Godot Engine Documentation

Release 2.1

Juan Linietsky, Ariel Manzur and the Godot community

May 04, 2020
Welcome to the official documentation of Godot Engine, the free and open source community-driven 2D and 3D game engine! If you are new to this documentation, we recommend that you read the introduction page to get an overview of what this documentation has to offer.

The table of contents below and in the sidebar should let you easily access the documentation for your topic of interest. You can also use the search function in the top left corner.

Note: Godot Engine is an open source project developed by a community of volunteers. It means that the documentation team can always use your feedback and help to improve the tutorials and class reference. If you do not manage to understand something, or cannot find what you are looking for in the docs, help us make the documentation better by letting us know!

Submit an issue or pull request on the GitHub repository, or discuss with us on the #godotengine-devel IRC channel!

The main documentation for the site is organized into the following sections:
1.1 Introduction

Welcome to the official documentation of Godot Engine, the free and open source community-driven 2D and 3D game engine! Behind this mouthful, you will find a powerful but user-friendly tool that you can use to develop any kind of game, for any platform and with no usage restriction whatsoever.

This page aims at giving a broad presentation of the engine and of the contents of this documentation, so that you know where to start if you are a beginner or where to look if you need info on a specific feature.

1.1.1 About Godot Engine

A game engine is a complex tool, and it is therefore difficult to present Godot in a few words. Here’s however our PR presentation, which you are free to reuse if you need a quick writeup about Godot Engine.

Godot Engine is a feature-packed, cross-platform game engine to create 2D and 3D games from a unified interface. It provides a comprehensive set of common tools, so that users can focus on making games without having to reinvent the wheel. Games can be exported in one click to a number of platforms, including the major desktop platforms (Linux, Mac OSX, Windows) as well as mobile (Android, iOS) and web-based (HTML5) platforms.

Godot is completely free and open source under the very permissive MIT license. No strings attached, no royalties, nothing. Users’ games are theirs, down to the last line of engine code. Godot’s development is fully independent and community-driven, empowering users to help shape their engine to match their expectations. It is supported by the Software Freedom Conservancy not-for-profit.

For a more in-depth view of the engine, you are encouraged to read this documentation further, especially the Step by step tutorial.

1.1.2 About the documentation

This documentation is continuously written, corrected, edited and revamped by members of the Godot Engine community. It is edited via text files in the reStructuredText markup language and then compiled into a static website/offline document using the open source Sphinx and ReadTheDocs tools.
Godot Engine Documentation, Release 2.1

Note: You can contribute to Godot’s documentation by opening issue tickets or sending patches via pull requests on its GitHub source repository.

All the contents are under the permissive Creative Commons Attribution 3.0 (CC-BY 3.0) license, with attribution to “Juan Linietsky, Ariel Manzur and the Godot Engine community”.

1.1.3 Organisation of the documentation

This documentation is organised in five sections with an impressively unbalanced distribution of contents – but the way it is split up should be relatively intuitive:

- The General section contains this introduction as well as information about the engine, its history, its licensing, authors, etc. It also contains the Frequently asked questions.
- The Learning section is the main raison d'être of this documentation, as it contains all the necessary information on using the engine to make games. It starts with the Step by step tutorial which should be the entry point for all new users. Its cornerstone is the Engine features category, which contains many feature-specific tutorials and documentation which can be read as needed, in any order.
- The Development section is intended for advanced users and contributors to the engine development, with information on compiling the engine, developing C++ modules or editor plugins.
- The Community gives information related to contributing to the engine development and the life of its community, e.g. how to report bugs, help with the documentation, etc. It’s also the place for tutorials contributed by community members before they are properly reviewed and adapted to be integrated in the official list of the Learning section.
- Finally, the Class reference is the documentation of the Godot API, which is also available directly within the script editor. It is generated automatically from a file in the main repository, and the generated files of the documentation are therefore not meant to be modified. See Updating the class reference for details.

Have fun reading and making games with Godot Engine!

1.2 Frequently asked questions

1.2.1 GDScript? Why your own scripting language? Why not Lua, Javascript, C#, etc.?

The short answer is, we’d rather a programmer does the small effort to learn GDScript so he or she later has a seamless experience, than attracting him or her with a familiar programming language that results in a worse experience. We are OK if you would rather not give Godot a chance because of this, but we strongly encourage you to try it and see the benefits yourself.

The official languages for Godot are GDScript and C++.

GDScript is designed to integrate from the ground to the way Godot works, more than any other language, and is very simple and easy to learn. Takes at most a day or two to get comfortable and it’s very easy to see the benefits once you do. Please do the effort to learn GDScript, you will not regret it.

Godot C++ API is also efficient and easy to use (the entire Godot editor is made with this API), and an excellent tool to optimize parts of a project, but trying to use it instead of GDScript for an entire game is, in most cases, a waste of time.
Yes, for more than a decade we tried in the past integrating several VMs (and even shipped games using them), such as Python, Squirrel and Lua (in fact we authored tolua++ in the past, one of the most popular C++ binders). None of them worked as well as GDScript does now.

More information about getting comfortable with GDScript or dynamically typed languages can be found in the GDScript more efficiently tutorial.

For the more technically versed, proceed to the next item.

1.2.2 I don’t believe you. What are the technical reasons for the item above?

The main reasons are:

1. No good thread support in most script VMs, and Godot uses threads (Lua, Python, Squirrel, JS, AS, etc.)
2. No good class extending support in most script VMs, and adapting to the way Godot works is highly inefficient (Lua, Python, JS)
3. Horrible interface for binding to C++, results in large amount of code, bugs, bottlenecks and general inefficiency (Lua, Python, Squirrel, JS, etc.)
4. No native vector types (vector3, matrix4, etc.), resulting in highly reduced performance when using custom types (Lua, Python, Squirrel, JS, AS, etc.)
5. Garbage collector results in stalls or unnecessarily large memory usage (Lua, Python, JS, AS, etc.)
6. Difficulty to integrate with the code editor for providing code completion, live editing, etc. (all of them). This is very well supported by GDScript.

GDScript was designed to solve the issues above, and performs very well in all the above scenarios. Please learn GDScript, and enjoy from a very smooth integration of scripting with the game engine (yes, it’s a rare but very enjoyable situation when things just work). It’s worth it, give it a try!

1.2.3 Why is FBX not supported for import?

FBX SDK has a very restrictive license, that is incompatible with the open license provided by Godot.

That said, Godot’s Collada support is really good, please use the OpenCollada exporter for maximum compatibility if you are using Maya or 3DS Max. If you are using Blender, take a look at our own Better Collada Exporter.

1.2.4 Will [Insert closed SDK such as PhysX, GameWorks, etc.] be supported in Godot?

No, the aim of Godot is to create a complete open source engine licensed under MIT, so you have complete control over every single piece of it. Open versions of functionality or features from such SDKs may be eventually added though.

That said, because it is open source, and modular, nothing prevents you or anyone else interested into adding those libraries as a module and ship your game using them, as either open or closed source. Everything is allowed.

1.2.5 How should assets be created to handle multiple resolutions and aspect ratios?

This question pops up often and it’s probably thanks to the misunderstanding created by Apple when they originally doubled the resolution of their devices. It made people think that having the same assets in different resolutions was a good idea, so many continued towards that path. That originally worked to a point and only for Apple devices, but
then several Android and Apple devices with different resolutions and aspect ratios were created, with a very wide range of sizes an DPIs.

The most common and proper way to this is to, instead, is to use a single base resolution for the game and only handle different screen aspects. This is mostly needed for 2D, as in 3D it’s just a matter of Cameara XFov or YFov.

1. Choose a single base resolution for your game. Even if there are devices that go up to 2K and devices that go down to 400p, regular hardware scaling in your device will take care of this at little or no performance cost. Most common choices are either near 1080p (1920x1080) or 720p (1280x720). Keep in mind the higher the resolution, the larger your assets, the more memory they will take and the longer the time it will take for loading.

2. Use the stretch options in Godot, 2D stretching with keeping aspect works best. Check the Multiple resolutions tutorial on how to achieve this.

3. Determine a minimum resolution and then decide if you want your game to stretch vertically or horizontally for different aspect ratios, or whether there is a minimum one and you want black bars to appear instead. This is also explained in the previous step.

4. For user interfaces, use the anchoring to determine where controls should stay and move. If UIs are more complex, consider learning about Containers.

And that’s it! Your game should work in multiple resolutions.

If there really is a desire to make your game also work on ancient devices with tiny screens (less than 300 pixels in width), you can use the export option to shrink images, and set that build to be used for certain screen sizes in the App Store or Google Play.

### 1.2.6 I have a great idea that will make Godot better, What do you think?

Your idea will most certainly be ignored. Examples of stuff that is ignored by the developers:

- Let’s do this because it will make Godot better
- Let’s do this in Godot because another game engine does it
- Let’s remove this because I think it’s not needed
- Let’s remove clutter and bloat and make Godot look nicer
- Let’s add an alternative workflow for people who prefer it

Developers are always willing to talk to you and listen to your feedback very openly, to an extent rarely seen in open source projects, but they will care mostly about real issues you have while using Godot, not ideas solely based on personal belief. Developers are interested in (for example):

- Your experience using the software and the problems you have, (we care about this much more than ideas on how to improve it)
- The features you would like to see implemented because you need them for your project.
- The concepts that were difficult to understand in order to learn the software.
- The parts of your workflow you would like to see optimized.

Once one of the above points is stated, we can work together on a solution and this is where your ideas and suggestions are most valuable and welcome, they need to be in context of a real issue.

As such, please don’t feel that your ideas for Godot are unwelcome. Instead, try to reformulate them as a problem first, so developers and the community have a base ground to discuss first.

Examples of how NOT to state problems generally are like this:

- Certain feature is ugly
• Certain workflow is slow
• Certain feature needs optimization
• Certain aspect of the UI looks cluttered

Associating something with an adjective will not get you much attention and developers will most likely not understand you. Instead, try to reformulate your problem as a story such as:

• I try to move objects around but always end up picking the wrong one
• I tried to make a game like Battlefield but I’m not managing to understand how to get lighting to look the same.
• I always forget which script I was editing, and it takes me too many steps to go back to it.

This will allow you to convey what you are thinking much better and set a common ground for discussion. Please try your best to state your problems as stories to the developers and the community, before discussing any idea.
2.1 Scenes and nodes

2.1.1 Introduction

Imagine for a second that you are not a game developer anymore. Instead, you’re a chef! Change your hipster outfit for a toque and a double breasted jacket. Now, instead of making games, you create new and delicious recipes for your guests.

So, how does a chef create a recipe? Recipes are divided into two sections: the first is the ingredients and the second is the instructions to prepare it. This way, anyone can follow the recipe and savor your magnificent creation.

Making games in Godot feels pretty much the same way. Using the engine feels like being in a kitchen. In this kitchen, nodes are like a refrigerator full of fresh ingredients to cook with.

There are many types of nodes, some show images, others play sound, other nodes display 3D models, etc. There’s dozens of them.

2.1.2 Nodes

But let’s start with the basics. A node is a basic element for creating a game, it has the following characteristics:

- Has a name.
- Has editable properties.
- Can receive a callback to process every frame.
- Can be extended (to have more functions).
- Can be added to other nodes as children.

The last one is very important. Nodes can have other nodes as children. When arranged in this way, the nodes become a **tree**.

In Godot, the ability to arrange nodes in this way creates a powerful tool for organizing projects. Since different nodes have different functions, combining them allows for creation of more complex functions.

This is probably not clear yet and makes little sense, but everything will click a few sections ahead. The most important fact to remember for now is that nodes exist and can be arranged this way.

### 2.1.3 Scenes

Now that the concept of nodes has been defined, the next logical step is to explain what a Scene is.

A scene is composed of a group of nodes organized hierarchically (in tree fashion). It has the following properties:

- A scene always has only one root node.
- Scenes can be saved to disk and loaded back.
- Scenes can be *instanced* (more on that later).
- Running a game means running a scene.
- There can be several scenes in a project, but for it to start, one of them must be selected to be loaded first.

Basically, the Godot editor is a **scene editor**. It has plenty of tools for editing 2D and 3D scenes as well as user interfaces, but the editor is based on the concept of editing a scene and the nodes that compose it.
2.1.4 Creating a new project

Theory is boring, so let’s change the subject and go practical. Following a long tradition in tutorials, the first project will be a Hello World. For this, the editor will be used.

When the godot executable is run outside a project, the Project Manager appears. This helps developers manage their projects.

To create a new project, the “New Project” option must be used. Choose and create a path for the project and specify the project name:
2.1.5 Editor

Once the “New Project” is created, the next step is opening it. This will open the Godot editor. Here is how the editor looks when freshly opened:

As mentioned before, making games in Godot feels like being in a kitchen, so let’s open the refrigerator and add some fresh nodes to the project. We’ll begin with a Hello World! To do this, the “New Node” button must be pressed (looks like a plus symbol underneath the “Scene” tab):

This will open the Create Node dialog, showing the long list of nodes that can be created:
From there, select the “Label” node first. It is under the “Control” node, but searching for it is probably the quickest way:

And finally, create the Label! A lot happens when Create is pressed:
First of all, the scene is changed to the 2D editor (because Label is a 2D Node type), and the Label appears, selected, at the top left corner of the viewport.

The node appears in the scene tree editor (box in the top left corner), and the label properties appear in the Inspector (box on the right side).

The next step will be to change the “Text” Property of the label, let’s change it to “Hello, World!”:

Ok, everything’s ready to run the scene! Press the PLAY SCENE Button on the top bar (or hit F6):

Aaaand... Oops.
Scenes need to be saved to be run, so save the scene to something like hello.scn in Scene -> Save:

And here’s when something funny happens. The file dialog is a special file dialog, and only allows you to save inside the project. The project root is “res://” which means “resource path”. This means that files can only be saved inside the project. For the future, when doing file operations in Godot, remember that “res://” is the resource path, and no matter the platform or install location, it is the way to locate where resource files are from inside the game.

After saving the scene and pressing run scene again, the “Hello, World!” demo should finally execute:

Success!

2.1.6 Configuring the project

Ok, it’s time to do some configuration to the project. Right now, the only way to run something is to execute the current scene. Projects, however, have several scenes so one of them must be set as the main scene. This scene is the one that will be loaded at the time the project is run.

These settings are all stored in the engine.cfg file, which is a plaintext file in win.ini format, for easy editing. There are dozens of settings that can be changed in this file to alter how a project executes, so to make matters simpler, a project setting dialog exists, which is sort of a frontend to editing engine.cfg
To access that dialog, simply go to Scene -> Project Settings.

Once the window opens, the task will be to select a main scene. This can be done easily by changing the application/main_scene property and selecting ‘hello.scn’.

With this change, pressing the regular Play button (or F5) will run the project, no matter which scene is being edited.

Going back to the project settings dialog. This dialog provides a lot of options that can be added to engine.cfg, and shows their default values. If the default value is ok, then there isn’t any need to change it.

When a value is changed, a tick is marked to the left of the name. This means that the property will be saved to the engine.cfg file and remembered.

As a side note, for future reference and a little out of context (this is the first tutorial after all!), it is also possible to add custom configuration options and read them in run-time using the \texttt{Globals} singleton.

\subsection*{2.1.7 To be continued...}

This tutorial talks about “scenes and nodes”, but so far there has been only \textit{one} scene and \textit{one} node! Don’t worry, the next tutorial will deal with that...

\section*{2.2 Instancing}

\subsection*{2.2.1 Rationale}

Having a scene and throwing nodes into it might work for small projects, but as a project grows, more and more nodes are used and it can quickly become unmanageable. To solve this, Godot allows a project to be separated into several scenes. This, however, does not work the same way as in other game engines. In fact, it’s quite different, so please do not skip this tutorial!

To recap: A scene is a collection of nodes organized as a tree, where they can have only one single node as the tree root.
In Godot, a scene can be created and saved to disk. As many scenes can be created and saved as desired.

Afterwards, while editing an existing or a new scene, other scenes can be instanced as part of it:

In the above picture, Scene B was added to Scene A as an instance. It may seem weird at first, but at the end of this tutorial it will make complete sense!

### 2.2.2 Instancing, step by step

To learn how to do instancing, let’s start with downloading a sample project: `instancing.zip`.

Unzip this project in any place of your preference. Then, add this project to the project manager using the ‘Import’ option:
Simply browse to inside the project location and open the “engine.cfg” file. The new project will appear on the list of projects. Edit the project by using the ‘Edit’ option.

This project contains two scenes “ball.scn” and “container.scn”. The ball scene is just a ball with physics, while container scene has a nicely shaped collision, so balls can be thrown in there.
Open the container scene, then select the root node:

Afterwards, push the link shaped button, this is the instancing button!

Select the ball scene (ball.scn), the ball should appear at the origin (0,0), move it to the center of the scene, like this:
Press Play and Voila!
The instanced ball fell to the bottom of the pit.

### 2.2.3 A little more

There can be as many instances as desired in a scene, just try instancing more balls, or duplicating them (Ctrl-D or duplicate button):
Then try running the scene again:
Cool, huh? This is how instancing works.

### 2.2.4 Editing instances

Select one of the many copies of the balls and go to the property editor. Let’s make it bounce a lot more, so look for the Bounce parameter and set it to 1.0:

![Property Editor](image)

Grey “revert” button will appear. When this button is present, it means we modified a property in the instanced scene to override a specific value in this instance. Even if that property is modified in the original scene, the custom value will always overwrite it. Pressing the revert button will restore the property to the original value that came from the scene.

### 2.2.5 Conclusion

Instancing seems handy, but there is more to it than meets the eye! The next part of the instancing tutorial should cover the rest..
2.3 Instancing (continued)

2.3.1 Recap

Instancing has many handy uses. At a glance, with instancing you have:

- The ability to subdivide scenes and make them easier to manage.
- A more flexible alternative to prefabs (and much more powerful given instances work at many levels).
- A way to design more complex game flows or even UIs (UI Elements are nodes in Godot too).

2.3.2 Design language

But the real strong point of instancing scenes is that it works as an excellent design language. This is pretty much what makes Godot special and different to any other engine out there. The entire engine was designed from the ground up around this concept.

When making games with Godot, the recommended approach is to leave aside other design patterns such as MVC or Entity-Relationship diagrams and start thinking about games in a more natural way. Start by imagining the visible elements in a game, the ones that can be named not just by a programmer but by anyone.

For example, here’s how a simple shooter game can be imagined:

![Diagram of a simple shooter game](image)

It’s pretty easy to come up with a diagram like this for almost any kind of game. Just write down the elements that come to mind, and then the arrows that represent ownership.

Once this diagram exists, making a game is about creating a scene for each of those nodes, and using instancing (either by code or from the editor) to represent ownership.

Most of the time programming games (or software in general) is spent designing an architecture and fitting game components to that architecture. Designing based on scenes replaces that and makes development much faster and more straightforward, allowing you to concentrate on the game itself. Scene/Instancing based design is extremely efficient at saving a large part of that work, since most of the components designed map directly to a scene. This way, none or little architectural code is needed.

The following is a more complex example, an open-world type of game with lots of assets and parts that interact:
Make some rooms with furniture, then connect them. Make a house later, and use those rooms as the interior.

The house can be part of a citadel, which has many houses. Finally the citadel can be put on the world map terrain. Add also guards and other NPCs to the citadel by previously creating their scenes.

With Godot, games can grow as quickly as desired, as all it needs is for more scenes to be created and instanced. The editor UI is designed to be operated by non programmers too, so a typical team development process can involve 3D or 2D artists, level designers, game designers, animators, etc all working with the editor interface.

2.3.3 Information overload!

Do not worry too much, the important part of this tutorial is to create awareness of how scenes and instancing are used in real life. The best way to understand all this is to make some games.

Everything will become very obvious when put to practice, so please do not scratch your head and go on to the next tutorial!

2.4 Scripting

2.4.1 Introduction

Much has been said about tools that allow users to create video games without programming. It’s been a dream for many independent developers to create games without learning how to code. This need has been around for a long time, even inside companies, where game designers wish to have more control of the game flow.

Many products have been shipped promising a no-programming environment, but the result is often incomplete, too complex or inefficient compared to traditional code. As a result, programming is here to stay for a long time. In fact, the general direction in game engines has been to add tools that try to reduce the amount of code that needs to be written for specific tasks, to speed up development.

In that sense, Godot has taken some useful design decisions towards that goal. The first and most important is the scene system. The aim of it is not obvious at first, but works well later on. That is, to relieve programmers from the responsibility of architecting code.

When designing games using the scene system, the whole project is fragmented into complementary scenes (not individual ones). Scenes complement each other, instead of being separate. There will be plenty of examples about this later on, but it’s very important to remember it.

For those with a good amount of programming expertise, this means a different design pattern to MVC. Godot promises efficiency at the expense of dropping the MVC habits, which are replaced by the scenes as a complement pattern.
Godot also uses the `extend` pattern for scripting, meaning that scripts extend from all the available engine classes.

### 2.4.2 GDScript

**GDScript** is a dynamically typed scripting language to fit inside Godot. It was designed with the following goals:

- First and most importantly, making it simple, familiar and as easy to learn as possible.
- Making the code readable and error safe. The syntax is mostly borrowed from Python.

Programmers generally take a few days to learn it, and within two weeks feel comfortable with it.

As with most dynamically typed languages, the higher productivity (code is easier to learn, faster to write, no compilation, etc) is balanced with a performance penalty. But most critical code is written in C++ already in the engine (vector ops, physics, math, indexing, etc), resulting in a more than sufficient performance for most types of games.

In any case, if more performance is required, critical sections can be rewritten in C++ and exposed transparently to the script. This allows the replacement of a GDScript class with a C++ class without altering the rest of the game.

### 2.4.3 Scripting a scene

Before continuing, please make sure to read the **GDScript** reference. It's a simple language and the reference is short, it will not take more than a few minutes to get an overview of the concepts.

#### Scene setup

This tutorial will begin by scripting a simple GUI scene. Use the add node dialog to create the following hierarchy, with the following nodes:

- Panel
  - Label
  - Button

It should look like this in the scene tree:

![Scene Tree Image]

Use the 2D editor to position and resize the button and label so that they look like the image below. You can set the text in the Inspector pane.
Finally, save the scene, a fitting name could be “sayhello.scn”

Adding a script

Right click on the panel node, then select “Add Script” in the context menu:

The script creation dialog will pop up. This dialog allows to select the language, class name, etc. GDScript does not use class names in script files, so that field is not editable. The script should inherit from “Panel” (as it is meant to extend the node, which is of Panel type, this is automatically filled).

Enter a path name for the script and then select “Create”:  

2.4. Scripting
Once this is done, the script will be created and added to the node. You can see this both as an extra icon in the node, as well as in the script property:
To edit the script, select either of the highlighted buttons. This will bring you to the script editor where an existing template will be included by default:
There is not much in there. The “_ready()” function is called when the node (and all its children) entered the active scene. (Remember, it’s not a constructor, the constructor is “_init()”).

**The role of the script**

A script adds behavior to a node. It is used to control the node functions as well as other nodes (children, parent, siblings, etc). The local scope of the script is the node (just like in regular inheritance) and the virtual functions of the node are captured by the script.

**Handling a signal**

Signals are used mostly in GUI nodes (although other nodes have them too). Signals are “emitted” when some specific kind of action happens, and can be connected to any function of any script instance. In this step, the “pressed” signal from the button will be connected to a custom function.

An interface for connecting signals to your scripts exists in the editor. You can access this by selecting the node in the scene tree and then selecting the “Node” tab. Make sure that you have “Signals” selected.
In any case, at this point it is clear that we are interested in the “pressed” signal. Instead of using the visual interface, we will opt to code the connection.

For this, a function exists that is probably the one most used by Godot programmers, namely `Node.get_node()`. This function uses paths to fetch nodes in the current tree or anywhere in the scene, relative to the node holding the script.

To fetch the button, the following must be used:

```godot
get_node("Button")
```

Next, a callback will be added that will change the label’s text when the button is pressed:

```godot
func _on_button_pressed():
    get_node("Label").set_text("HELLO!")
```

Finally, the button “pressed” signal will be connected to that callback in `_ready()`, by using `Object.connect()`.

```godot
func _ready():
    get_node("Button").connect("pressed", self, "_on_button_pressed")
```

The final script should look like this:

```godot
extends Panel

# member variables here, example:
# var a=2
# var b="textvar"
```
func _on_button_pressed():
    get_node("Label").set_text("HELLO!")

func _ready():
    get_node("Button").connect("pressed", self, "_on_button_pressed")

Running the scene should have the expected result when pressing the button:

Note: As it is a common misunderstanding in this tutorial, let’s clarify again that get_node(path) works by returning the immediate children of the node controlled by the script (in this case, Panel), so Button must be a child of Panel for the above code to work. To give this clarification more context, if Button were a child of Label, the code to obtain it would be:

```python
# not for this case
# but just in case
get_node("Label/Button")
```

Also, remember that nodes are referenced by name, not by type.

---

2.5 Scripting (continued)

2.5.1 Processing

Several actions in Godot are triggered by callbacks or virtual functions, so there is no need to check for writing code that runs all the time. Additionally, a lot can be done with animation players.

However, it is still a very common case to have a script process on every frame. There are two types of processing, idle processing and fixed processing.

Idle processing is activated with the Node.set_process() function. Once active, the Node._process() callback will be called every frame. Example:

```python
func _ready():
    set_process(true)

func _process(delta):
    # do something...
```

The delta parameter describes the time elapsed (in seconds, as floating point) since the previous call to _process(). Fixed processing is similar, but only needed for synchronization with the physics engine.
A simple way to test this is to create a scene with a single Label node, with the following script:

```godot
extends Label
var accum=0
func _ready():
  set_process(true)
func _process(delta):
  accum += delta
  set_text(str(accum))
```

Which will show a counter increasing each frame.

### 2.5.2 Groups

Nodes can be added to groups (as many as desired per node). This is a simple yet useful feature for organizing large scenes. There are two ways to do this, the first is from the UI, from the Groups button under Node-panel:

![Groups UI](image)

And the second from code. One useful example would be to tag scenes which are enemies.

```godot
func _ready():
  add_to_group("enemies")
```

This way, if the player is discovered sneaking into the secret base, all enemies can be notified about the alarm sounding, by using `SceneTree.call_group()`:

```godot
func _on_discovered():
  get_tree().call_group(0, "guards", "player_was_discovered")
```

The above code calls the function “player_was_discovered” on every member of the group “guards”.

Optionally, it is possible to get the full list of “guards” nodes by calling `SceneTree.get_nodes_in_group()`:

```godot
var guards = get_tree().get_nodes_in_group("guards")
```

More will be added about `SceneTree` later.

### 2.5.3 Notifications

Godot has a system of notifications. This is usually not needed for scripting, as it’s too low level and virtual functions are provided for most of them. It’s just good to know they exist. Simply add a `Object._notification()` function in your script:

```godot
```
The documentation of each class in the *Class Reference* shows the notifications it can receive. However, for most cases GDScript provides simpler overrideable functions.

### 2.5.4 Overrideable functions

Nodes provide many useful overrideable functions, which are described as follows:

```godot
def _notification(what):
    if (what == NOTIFICATION_READY):
        print("This is the same as overriding _ready()...")
    elif (what == NOTIFICATION_PROCESS):
        var delta = get_process_time()
        print("This is the same as overriding _process()...")
```

As mentioned before, it’s best to use these functions.
### 2.5.5 Creating nodes

To create a node from code, just call the .new() method (like for any other class based datatype). Example:

```gdscript
var s
func _ready():
    s = Sprite.new() # create a new sprite!
    add_child(s) # add it as a child of this node
```

To delete a node, be it inside or outside the scene, free() must be used:

```gdscript
func _someaction():
    s.free() # immediately removes the node from the scene and frees it
```

When a node is freed, it also frees all its children nodes. Because of this, manually deleting nodes is much simpler than it appears. Just free the base node and everything else in the sub-tree goes away with it.

However, it might happen very often that we want to delete a node that is currently “blocked”, because it is emitting a signal or calling a function. This will result in crashing the game. Running Godot in the debugger often will catch this case and warn you about it.

The safest way to delete a node is by using `Node.queue_free()`. This erases the node safely during idle.

```gdscript
func _someaction():
    s.queue_free() # remove the node and delete it while nothing is happening
```

### 2.5.6 Instancing scenes

Instancing a scene from code is pretty easy and done in two steps. The first one is to load the scene from disk.

```gdscript
var scene = load("res://myscene.scn") # will load when the script is instanced
```

Preloading it can be more convenient sometimes, as it happens at parse time.

```gdscript
var scene = preload("res://myscene.scn") # will load when parsing the script
```

But ‘scene’ is not yet a node containing subnodes. It’s packed in a special resource called `PackedScene`. To create the actual node, the function `PackedScene.instance()` must be called. This will return the tree of nodes that can be added to the active scene:

```gdscript
var node = scene.instance()
add_child(node)
```

The advantage of this two-step process is that a packed scene may be kept loaded and ready to use, so it can be used to create as many instances as desired. This is especially useful to quickly instance several enemies, bullets, etc. in the active scene.

### 2.6 Simple 2D game

#### 2.6.1 Pong

In this tutorial, a basic game of Pong will be created. There are plenty of more complex examples in the demos included with the engine, but this should get you introduced to the basic functionalities for 2D Games.

To begin with, run the Godot Engine and start a new project.
2.6.2 Assets

Some assets are included for this tutorial: pong_assets.zip. Unzip its content in your project folder.

2.6.3 Scene setup

For the sake of the old times, the game will be in 640x400 pixels resolution. This can be configured in the Project Settings (see Configuring the project) under Scene/Project settings menu. The default background color should be set to black:

Create a Node2D node for the project root. Node2D is the base type for the 2D engine. After this, add some sprites (Sprite node) for the left and right paddles, the separator and ball. You can set a custom name for each node, and set the texture for each sprite in the Inspector.

Set nodes positions:

- “left” node: (67, 183)
- “right” node: (577, 187)
- “separator” node: (320, 200)
- “ball” node: (320, 188)

The final scene layout should look similar to this (note: the ball is in the middle!):
Save the scene as “pong.tscn” and set it as the main scene in the project properties.

### 2.6.4 Input actions setup

Video games can be played using various input methods: Keyboard, Joypad, Mouse, Touchscreen (multitouch)...

Godot is able to use them all. However, it would be interesting to define the inputs as “Input Actions” instead of hardware actions that you’d manage separately. This way, any input method can be used: each of them only require the user to connect buttons to game actions that you defined.

This is Pong. The only input that matters is for the pads going up and down.

Open the project properties dialog again (Scene/Project settings), but this time move to the “Input Map” tab.

In this tab, add 4 actions: left_move_up, left_move_down, right_move_up, right_move_down. Assign the keys that you desire. AI/Z (for the left player) and Up/Down (for the right player) as keys should work in most cases.
Create a script for the root node of the scene and open it (as explained in Adding a script). This script inherits Node2D:

```godot
extends Node2D
func _ready():
    pass
```

First things first, we need to define some members for our script so it can store useful values. Such values are the dimensions of the screen, the pad and the initial direction of the ball.

```godot
extends Node2D
# Member variables
var screen_size
var pad_size
var direction = Vector2(1.0, 0.0)
func _ready():
    pass
```

As you know, the _ready() function is the first function called (after _enter_tree() which we don’t need here). In this function, two things have to be done. The first one is to enable processing: this is the purpose of the set_process(true) function. The second one is to initialize our two member variables.
We initialize the `pad_size` variable by getting one of the pads nodes (the left one here), and obtain its texture size. The `screen_size` is initialized using the `get_viewport_rect()` which returns a Rect object corresponding to the game window, and we store its size.

Now, we need to add some other members to our script in order to make our ball move.

```
extends Node2D

# Member variables
var screen_size
var pad_size
var direction = Vector2(1.0, 0.0)

# Constant for ball speed (in pixels/second)
const INITIAL_BALL_SPEED = 80
# Speed of the ball (also in pixels/second)
var ball_speed = INITIAL_BALL_SPEED
# Constant for pads speed
const PAD_SPEED = 150

func _ready():
    screen_size = get_viewport_rect().size
    pad_size = get_node("left").get_texture().get_size()
    set_process(true)
```

Finally, the `_process()` function. All the code below is contained by this function.

We have to init some useful values for computation. The first one is the ball position (from the node), the second one is the rectangle (Rect2) for each pad. These rectangles will be used for collision tests between the ball and the pads. Sprites center their textures by default, so a small adjustment of `pad_size / 2` must be added.

```
func _process(delta):
    var ball_pos = get_node("ball").get_pos()
    var left_rect = Rect2( get_node("left").get_pos() - pad_size*0.5, pad_size )
    var right_rect = Rect2( get_node("right").get_pos() - pad_size*0.5, pad_size )
```

Now, let’s add some movement to the ball in the `_process()` function. Since the ball position is stored in the `ball_pos` variable, integrating it is simple:

```
# Integrate new ball position
ball_pos += direction * ball_speed * delta
```

This code line is called at each iteration of the `_process()` function. That means the ball position will be updated at each new frame.

Now that the ball has a new position, we need to test if it collides with anything, that is the window borders and the pads. First, the floor and the roof:

```
# Flip when touching roof or floor
if ( (ball_pos.y < 0 and direction.y < 0) or (ball_pos.y > screen_size.y and direction.y > 0)):
```
direction.y = -direction.y

Second, the pads: if one of the pads is touched, we need to invert the direction of the ball on the X axis so it goes back, and define a new random Y direction using the `randf()` function. We also increase its speed a little.

```godot
# Flip, change direction and increase speed when touching pads
if ((left_rect.has_point(ball_pos) and direction.x < 0) or (right_rect.has_point(ball_pos) and direction.x > 0)):
    direction.x = -direction.x
    direction.y = randf() * 2.0 - 1
    direction = direction.normalized()
    ball_speed *= 1.1
```

Finally, if the ball went out of the screen, it’s game over. That is, we test if the X position of the ball is less than 0 or greater than the screen width. If so, the game restarts:

```godot
# Check gameover
if (ball_pos.x < 0 or ball_pos.x > screen_size.x):
    ball_pos = screen_size * 0.5
    ball_speed = INITIAL_BALL_SPEED
    direction = Vector2(-1, 0)
```

Once everything is done, the node is updated with the new position of the ball, which was computed before:

```godot
get_node("ball").set_pos(ball_pos)
```

Next, we allow the pads to move. We only update their position according to player input. This is done using the Input class:

```godot
# Move left pad
var left_pos = get_node("left").get_pos()

if (left_pos.y > 0 and Input.is_action_pressed("left_move_up")):  
    left_pos.y += -PAD_SPEED * delta

if (left_pos.y < screen_size.y and Input.is_action_pressed("left_move_down")):  
    left_pos.y += PAD_SPEED * delta

get_node("left").set_pos(left_pos)

# Move right pad
var right_pos = get_node("right").get_pos()

if (right_pos.y > 0 and Input.is_action_pressed("right_move_up")):  
    right_pos.y += -PAD_SPEED * delta

if (right_pos.y < screen_size.y and Input.is_action_pressed("right_move_down")):  
    right_pos.y += PAD_SPEED * delta

get_node("right").set_pos(right_pos)
```

We use the four actions previously defined in the Input actions setup section above. When the player activates the respective key, the corresponding action is triggered. As soon as this happens, we simply compute a new position for the pad in the desired direction and apply it to the node.

That’s it! A simple Pong was written with a few lines of code.
2.7 GUI tutorial

2.7.1 Introduction

If there is something that most programmers hate with passion, it is programming graphical user interfaces (GUIs). It's boring, tedious and unchallenging. Several aspects make matters worse, such as:

- Pixel alignment of UI elements is difficult (such that it looks just like the designer intends).
- UIs are changed constantly due to design and usability issues that appear during testing.
- Handling proper screen re-sizing for different display resolutions.
- Animating several screen components, to make it look less static.

GUI programming is one of the leading causes of programmer burnout. During the development of Godot (and previous engine iterations), several techniques and philosophies for UI development were put into practice, such as immediate mode, containers, anchors, scripting, etc. This was always done with the main goal of reducing the stress programmers had to face while putting together user interfaces.

In the end, the resulting UI subsystem in Godot is an efficient solution to this problem, and works by mixing together a few different approaches. While the learning curve is a little steeper than in other toolkits, developers can put together complex user interfaces in very little time, by sharing the same set of tools with designers and animators.

2.7.2 Control

The basic node for UI elements is `Control` (sometimes called “Widget” or “Box” in other toolkits). Every node that provides user interface functionality descends from it.

When controls are put in a scene tree as a child of another control, its coordinates (position, size) are always relative to the parent. This sets the basis for editing complex user interfaces quickly and visually.

2.7.3 Input and drawing

Controls receive input events by means of the `Control._input_event()` callback. Only one control, the one in focus, will receive keyboard/joypad events (see `Control.set_focus_mode()` and `Control.grab_focus()`).

Mouse motion events are received by the control directly below the mouse pointer. When a control receives a mouse button pressed event, all subsequent motion events are received by the pressed control until that button is released, even if the pointer moves outside the control boundary.

Like any class that inherits from `CanvasItem` (Control does), a `CanvasItem._draw()` callback will be received at the beginning and every time the control needs to be redrawn (the programmer needs to call `CanvasItem.update()` to enqueue the CanvasItem for redraw). If the control is not visible (yet another CanvasItem property), the control does not receive any input.

In general though, the programmer does not need to deal with drawing and input events directly when building UIs (that is more useful when creating custom controls). Instead, controls emit different kinds of signals with contextual information for when action occurs. For example, a `Button` emits a “pressed” signal when pressed, a `Slider` will emit a “value_changed” when dragged, etc.

2.7.4 Custom control mini tutorial

Before going into more depth, creating a custom control will be a good way to get the picture on how controls works, as they are not as complex as it might seem.
Additionally, even though Godot comes with dozens of controls for different purposes, it happens often that it’s easier to attain a specific functionality by creating a new one.

To begin, create a single-node scene. The node is of type “Control” and has a certain area of the screen in the 2D editor, like this:

Add a script to that node, with the following code:

```godot
extends Control

var tapped=false

func _draw():
    var r = Rect2( Vector2(), get_size() )
    if (tapped):
        draw_rect(r, Color(1,0,0) )
    else:
        draw_rect(r, Color(0,0,1) )

func _input_event(ev):
    if (ev.type==InputEvent.MOUSE_BUTTON and ev.pressed):
        tapped=true
        update()
```

Then run the scene. When the rectangle is clicked/tapped, it will change color from blue to red. That synergy between the events and the drawing is pretty much how most controls work internally.
2.7.5 UI complexity

As mentioned before, Godot includes dozens of controls ready for use in a user interface. Such controls are divided into two categories. The first is a small set of controls that work well for creating most game user interfaces. The second (and most controls are of this type) are meant for complex user interfaces and uniform skinning through styles. A description is presented as follows to help understand which one should be used in which case.

2.7.6 Simplified UI controls

This set of controls is enough for most games, where complex interactions or ways to present information are not necessary. They can be skinned easily with regular textures.

- **Label**: Node used for showing text.
- **TextureFrame**: Displays a single texture, which can be scaled or kept fixed.
- **TextureButton**: Displays a simple textured button, states such as pressed, hover, disabled, etc. can be set.
- **TextureProgress**: Displays a single textured progress bar.

Additionally, re-positioning of controls is most efficiently done with anchors in this case (see the Size and anchors tutorial for more information).

In any case, it will happen often that even for simple games, more complex UI behaviors are required. An example of this is a scrolling list of elements (for a high score table, for example), which needs a `ScrollContainer` and a `VBoxContainer`. These kind of more advanced controls can be mixed with the regular ones seamlessly (they are all controls after all).

2.7.7 Complex UI controls

The rest of the controls (and there are dozens of them!) are meant for another set of scenarios, most commonly:

- Games that require complex UIs, such as PC RPGs, MMOs, strategy, sims, etc.
- Creating custom development tools to speed up content creation.
- Creating Godot Editor Plugins, to extend the engine functionality.

Re-positioning controls for these kind of interfaces is more commonly done with containers (see the Size and anchors tutorial for more information).
2.8 Splash screen

2.8.1 Tutorial

This is a simple tutorial to establish the basic idea of how the GUI subsystem works. The goal is to create a really simple, static splash screen.

Following is a file with the assets that will be used. These can be added directly to your project folder - no need to import them:

robisplash_assets.zip.

2.8.2 Setting up

Set the display resolution to 800x450 in Project Settings, and set up a new scene like this:

![Scene setup](image)

The nodes “background” and “logo” are of TextureFrame type. These have a special property for setting the texture to be displayed, just load the corresponding file.
The node “start” is a TextureButton. It takes several images for different states, but only the normal and pressed will be supplied in this example:

Finally, the node “copyright” is a Label. A custom font can be set for labels by editing the following property:
As a side note, the font was imported from a TTF, see Importing fonts.

2.9 Animations

2.9.1 Introduction

This tutorial will explain how everything is animated in Godot. Godot’s animation system is extremely powerful and flexible.

To begin, let’s just use the scene from the previous tutorial (Splash screen). The goal is, to add a simple animation to it. Here’s a copy just in case: robisplash.zip.

2.9.2 Creating the animation

First of all, add an AnimationPlayer node to the scene as a child of “bg” (the root node):
When a node of this type is selected, the animation editor panel will appear:

So, it’s time to create a new animation! Press the new animation button and name the animation “intro”.

After the animation has been created, it’s time to edit it, by pressing the “edit” button:

2.9.3 Editing the animation

Now this is when the magic happens! Several things happen when the “edit” button is pressed, the first one is that the animation editor appears above the animation panel. (In Godot 2.x, this button has been removed, instead, click on the ‘animation’ toggle at the bottom right for similar functionality.)

The second and most important, is that the property editor enters “animation editing” mode. In this mode, a key icon
appears next to every property of the property editor. This means that, in Godot, *any property of any object* can be animated:

![Property Editor](image)

### 2.9.4 Making the logo appear

The logo will appear from the top of the screen. After selecting the animation player, the editor panel will stay visible until manually hidden (or the animation node is erased). Taking advantage of this, select the “logo” node and go to the “pos” property, move it up to position: 114,-400.

Once in this position, press the key button next to the property:
As the track is new, a dialog will appear asking to create it. Confirm it!

The keyframe will be added in the animation player editor:
Move the editor cursor to the end, by clicking here:

Change the logo position to 114,0 and add a keyframe again. With two keyframes, the animation happens.

Pressing Play on the animation panel will make the logo descend. To test it by running the scene, the autoplay button can tag the animation to start automatically when the scene starts:
And finally, when running the scene, the animation should look like this:

2.10 Resources

2.10.1 Nodes and resources

So far, Nodes have been the most important datatype in Godot, as most of the behaviors and features of the engine are implemented through them. There is another datatype that is equally important: Resource.

Where Nodes focus on behaviors, such as drawing a sprite, drawing a 3D model, physics, GUI controls, etc, Resources are mere data containers. This means that they don’t do any action nor process any information. Resources just contain data.

Examples of resources are Texture, Script, Mesh, Animation, Sample, AudioStream, Font, Translation, etc.

When Godot saves or loads (from disk) a scene (.scn or .xml), an image (png, jpg), a script (.gd) or pretty much anything, that file is considered a resource.

When a resource is loaded from disk, it is always loaded once. That means, if there is a copy of that resource already loaded in memory, trying to load the resource again will just return the same copy again and again. This corresponds with the fact that resources are just data containers, so there is no need to have them duplicated.

Typically, every object in Godot (Node, Resource, or anything else) can export properties, properties can be of many types (like a string, integer, Vector2, etc) and one of those types can be a resource. This means that both nodes and resources can contain resources as properties. To make it a little more visual:
2.10.2 External vs built-in

The resource properties can reference resources in two ways, external (on disk) or built-in.

To be more specific, here’s a Texture in a Sprite node:
Pressing the “>” button on the right side of the preview allows to view and edit the resources properties. One of the properties (path) shows where it comes from. In this case, it comes from a png image.

When the resource comes from a file, it is considered an external resource. If the path property is erased (or it never had a path to begin with), it is considered a built-in resource.

For example, if the path “res://robi.png” is erased from the “path” property in the above example, and then the scene is saved, the resource will be saved inside the .scn scene file, no longer referencing the external “robi.png”. However, even if saved as built-in, and even though the scene can be instanced multiple times, the resource will always be loaded
only once. That means, different Robi robot scenes instanced at the same time will still share the same image.

### 2.10.3 Loading resources from code

Loading resources from code is easy. There are two ways to do it. The first is to use load(), like this:

```godot
func _ready():
    var res = load("res://robi.png") # resource is loaded when line is executed
    get_node("sprite").set_texture(res)
```

The second way is more optimal, but only works with a string constant parameter, because it loads the resource at compile-time.

```godot
func _ready():
    var res = preload("res://robi.png") # resource is loaded at compile time
    get_node("sprite").set_texture(res)
```

### 2.10.4 Loading scenes

Scenes are also resources, but there is a catch. Scenes saved to disk are resources of type \textit{PackedScene}, this means that the scene is packed inside a resource.

To obtain an instance of the scene, the method \texttt{PackedScene.instance()} must be used.

```godot
func _on_shoot():
    var bullet = preload("res://bullet.scn").instance()
    add_child(bullet)
```

This method creates the nodes in hierarchy, configures them (sets all the properties) and returns the root node of the scene, which can be added to any other node.

The approach has several advantages. As the \texttt{PackedScene.instance()} function is pretty fast, adding extra content to the scene can be done efficiently. New enemies, bullets, effects, etc can be added or removed quickly, without having to load them again from disk each time. It is important to remember that, as always, images, meshes, etc are all shared between the scene instances.

### 2.10.5 Freeing resources

Resource extends from \textit{Reference}. As such, when a resource is no longer in use, it will automatically free itself. Since, in most cases, Resources are contained in Nodes, scripts or other resources, when a node is removed or freed, all the children resources are freed too.

### 2.10.6 Scripting

Like any object in Godot, not just nodes, resources can be scripted, too. However, there isn’t generally much of an advantage, as resources are just data containers.
2.11 File system

2.11.1 Introduction

File systems are yet another hot topic in engine development. The file system manages how the assets are stored, and how they are accessed. A well designed file system also allows multiple developers to edit the same source files and assets while collaborating together.

Initial versions of the Godot engine (and previous iterations before it was named Godot) used a database. Assets were stored in it and assigned an ID. Other approaches were tried as well, such as local databases, files with metadata, etc. In the end the simple approach won and now Godot stores all assets as files in the file system.

2.11.2 Implementation

The file system stores resources on disk. Anything, from a script, to a scene or a PNG image is a resource to the engine. If a resource contains properties that reference other resources on disk, the paths to those resources are also included. If a resource has sub-resources that are built-in, the resource is saved in a single file together with all the bundled sub-resources. For example, a font resource is often bundled together with the font textures.

In general the Godot file system avoids using metadata files. The reason for this is simple, existing asset managers and VCSs are simply much better than anything we can implement, so Godot tries the best to play along with SVN, Git, Mercurial, Perforce, etc.

Example of a file system contents:

```
/engine.cfg
/enemy/enemy.scn
/enemy/enemy.gd
/enemy/enemysprite.png
/player/player.gd
```

2.11.3 engine.cfg

The engine.cfg file is the project description file, and it is always found at the root of the project. In fact its location defines where the root is. This is the first file that Godot looks for when opening a project.

This file contains the project configuration in plain text, using the win.ini format. Even an empty engine.cfg can function as a basic definition of a blank project.

2.11.4 Path delimiter

Godot only supports / as a path delimiter. This is done for portability reasons. All operating systems support this, even Windows, so a path such as c:\project\engine.cfg needs to be typed as c:/project/engine.cfg.

2.11.5 Resource path

When accessing resources, using the host OS file system layout can be cumbersome and non-portable. To solve this problem, the special path res:// was created.

The path res:// will always point at the project root (where engine.cfg is located, so in fact res://engine.cfg is always valid).
This file system is read-write only when running the project locally from the editor. When exported or when running on different devices (such as phones or consoles, or running from DVD), the file system will become read-only and writing will no longer be permitted.

2.11.6 User path

Writing to disk is often still needed for various tasks such as saving game state or downloading content packs. To this end, the engine ensures that there is a special path `user://` that is always writable.

2.11.7 Host file system

Alternatively host file system paths can also be used, but this is not recommended for a released product as these paths are not guaranteed to work on all platforms. However, using host file system paths can be very useful when writing development tools in Godot!

2.11.8 Drawbacks

There are some drawbacks to this simple file system design. The first issue is that moving assets around (renaming them or moving them from one path to another inside the project) will break existing references to these assets. These references will have to be re-defined to point at the new asset location.

The second is that under Windows and OSX file and path names are case insensitive. If a developer working in a case insensitive host file system saves an asset as “myfile.PNG”, but then references it as “myfile.png”, it will work just fine on their platform, but not on other platforms, such as Linux, Android, etc. This may also apply to exported binaries, which use a compressed package to store all files.

It is recommended that your team clearly defines a naming convention for files when working with Godot! One simple fool-proof convention is to only allow lowercase file and path names.

2.12 SceneTree

2.12.1 Introduction

This is where things start getting abstract, but don’t panic, as there’s not really more depth than this.

In previous tutorials, everything revolves around the concept of nodes, scenes are made of them, and they become active once they enter the scene tree.

This concept deserves going into a little more detail. In fact, the scene system is not even a core component of Godot, as it is possible to skip it and write a script (or C++ code) that talks directly to the servers. But making a game that way would be a lot of work.

2.12.2 MainLoop

The way Godot works internally is as follows. There is the OS class, which is the only instance that runs at the beginning. Afterwards, all drivers, servers, scripting languages, scene system, etc are loaded.

When initialization is complete, OS needs to be supplied a MainLoop to run. Up to this point, all this is internals working (you can check main/main.cpp file in the source code if you are ever interested to see how this works internally).
The user program, or game, starts in the MainLoop. This class has a few methods, for initialization, idle (framesynchronized callback), fixed (physics-synchronized callback), and input. Again, this is really low level and when making games in Godot, writing your own MainLoop does not even make sense.

### 2.12.3 SceneTree

One of the ways to explain how Godot works, is that it’s a high level game engine over a low level middleware. The scene system is the game engine, while the OS and servers are the low level API.

In any case, the scene system provides its own main loop to OS, SceneTree. This is automatically instanced and set when running a scene, no need to do any extra work.

It’s important to know that this class exists because it has a few important uses:

- It contains the root Viewport, to which a scene is added as a child when it’s first opened, to become part of the Scene Tree (more on that next)
- It contains information about the groups, and has means to call all nodes in a group, or get a list of them.
- It contains some global state functionality, such as setting pause mode, or quitting the process.

When a node is part of the Scene Tree, the SceneTree singleton can be obtained by simply calling Node.get_tree().

### 2.12.4 Root viewport

The root Viewport is always at the top of the scene. From a node, it can be obtained in two different ways:

```python
get_tree().get_root()  # access via scenemainloop
get_node("/root")  # access via absolute path
```

This node contains the main viewport, anything that is a child of a Viewport is drawn inside of it by default, so it makes sense that the top of all nodes is always a node of this type, otherwise nothing would be seen!

While other viewports can be created in the scene (for split-screen effects and such), this one is the only one that is never created by the user. It’s created automatically inside SceneTree.

### 2.12.5 Scene tree

When a node is connected, directly or indirectly, to the root viewport, it becomes part of the scene tree.

This means that, as explained in previous tutorials, it will get the _enter_tree() and _ready() callbacks (as well as _exit_tree()).
When nodes enter the Scene Tree, they become active. They get access to everything they need to process, get input, display 2D and 3D, notifications, play sound, groups, etc. When they are removed from the scene tree, they lose access.

### 2.12.6 Tree order

Most node operations in Godot, such as drawing 2D, processing or getting notifications are done in tree order. This means that parents and siblings with a smaller rank in the tree order will get notified before the current node.

### 2.12.7 “Becoming active” by entering the Scene Tree

1. A scene is loaded from disk or created by scripting.
2. The root node of that scene (only one root, remember?) is added as either a child of the “root” Viewport (from SceneTree), or to any child or grand-child of it.
3. Every node of the newly added scene, will receive the “enter_tree” notification ( _enter_tree() callback in GDScript) in top-to-bottom order.
4. An extra notification, “ready” (_ready() callback in GDScript) is provided for convenience, when a node and all its children are inside the active scene.

5. When a scene (or part of it) is removed, they receive the “exit scene” notification (_exit_tree() callback in GDScript) in bottom-to-top order.

### 2.12.8 Changing current scene

After a scene is loaded, it is often desired to change this scene for another one. The simple way to do this is to use the `SceneTree.change_scene()` function:

```gd
func _my_level_was_completed():
    get_tree().change_scene("res://levels/level2.scn")
```

This is a quick and useful way to switch scenes, but has the drawback that the game will stall until the new scene is loaded and running. At some point in your game, it may be desired to create proper loading screens with progress bar, animated indicators or thread (background) loading. This must be done manually using autoloads (see next chapter!) and **Background loading**.

### 2.13 Singletons (AutoLoad)

#### 2.13.1 Introduction

Scene singletons are very useful, catering to a common use case where you need to store persistent information between scenes.

Albeit very powerful, the scene system by itself has a few drawbacks:

- There is no common place to store information (e.g. a player’s items etc.) required by more than one scene.
- While it is possible for a scene that loads and unloads other scenes as its children to store information common to these child scenes, it is no longer possible to run these scenes by themselves and expect them to work correctly.
- While information can be stored to disk in ‘user://’ and this information can be loaded by scenes that require it, continuously saving and loading this data when changing scenes is cumbersome and may be slow.

However there is still a need in Godot to create parts of a scene that:

- Are always loaded, no matter which scene is opened from the editor
- Can store global variables, such as player information, items, money etc. and share information between scenes
- Can handle switching scenes and transitions
- Acts like a singleton, since GDScript does not support global variables by design.

Auto-loading nodes and scripts caters to this need.

#### 2.13.2 AutoLoad

You can use AutoLoad to load a scene, or a script that inherits from Node (a node will be created and the script will be set to it).

To autoload a scene or script, select Scene > Project Settings from the menu and switch to the AutoLoad tab. Each entry in the list requires a name, which is used as the name of the node, and the node is always added to the root viewport before any other scenes are loaded.
2.13.3 Custom scene switcher

This short tutorial will explain how to make a scene switcher using autoload. For simple scene switching, the SceneTree.change_scene() method suffices (described in SceneTree), so this method is for more complex behavior when switching between scenes.

First download the template from here: autoload.zip, then open it.

Two scenes are present, scene_a.scn and scene_b.scn on an otherwise empty project. Each are identical and contain a button connected to a callback for switching to the other scene. When the project runs, it starts in scene_a.scn. However, this currently does nothing and pressing the button does not work.

2.13.4 global.gd

First of all, create a global.gd script. The easy way to create a resource from scratch is from the new resource button in the inspector tab:
Save the script as *global.gd*:

The script should open in the script editor. The next step is to add it to AutoLoad list. Select Scene > Project Settings from the menu, switch to the AutoLoad tab and add a new entry with name “global” that points to this file:
Now, whenever you run any of your scenes, the script is always loaded.

Returning to our script, the current scene needs to be fetched in the _ready() function. Both the current scene and global.gd are children of root, but the autoloaded nodes are always first. This means that the last child of root is always the loaded scene.

Note: Make sure that global.gd extends Node, otherwise it won’t be loaded!

```godot
extends Node

var current_scene = null
func _ready():
    var root = get_tree().get_root()
    current_scene = root.get_child( root.get_child_count() -1 )
```

Next up is the function for changing the scene. This function frees the current scene and replaces it with the requested one.

```godot
func goto_scene(path):
    # This function will usually be called from a signal callback,  
    # or some other function from the running scene.  
    # Deleting the current scene at this point might be  
    # a bad idea, because it may be inside of a callback or function of it.  
    # The worst case will be a crash or unexpected behavior.  
    # The way around this is deferring the load to a later time, when  
    # it is ensured that no code from the current scene is running:  
    
    call_deferred("_deferred_goto_scene",path)

func _deferred_goto_scene(path):
    # Immediately free the current scene,  
    # there is no risk here.  
    current_scene.free()
    
    # Load new scene  
    var s = ResourceLoader.load(path)
    # Instance the new scene
```
current_scene = s.instance()

# Add it to the active scene, as child of root
get_tree().get_root().add_child(current_scene)

# optional, to make it compatible with the SceneTree.change_scene() API
get_tree().set_current_scene( current_scene )

As mentioned in the comments above, we really want to avoid the situation of having the current scene being deleted while being used (code from functions of it being run), so using `Object.call_deferred()` is desired at this point. The result is that execution of the commands in the second function will happen at a later time when no code from the current scene is running.

Finally, all that is left is to fill the empty functions in scene_a.gd and scene_b.gd:

```gd
#add to scene_a.gd

func _on_goto_scene_pressed():
    get_node("/root/global").goto_scene("res://scene_b.scn")

```

and

```gd
#add to scene_b.gd

func _on_goto_scene_pressed():
    get_node("/root/global").goto_scene("res://scene_a.scn")

```

Now if you run the project, you can switch between both scenes by pressing the button!

To load scenes with a progress bar, check out the next tutorial, *Background loading*
3.1 2D and 3D keybindings

3.1.1 2D viewport
3.1.2 3D viewport

![3D viewport screenshot]

3.2 From Unity3D to Godot Engine

This guide provides an overview of Godot Engine from the viewpoint of a Unity user, and aims to help you migrate your existing Unity experience into the world of Godot.
### 3.2.1 Differences

<table>
<thead>
<tr>
<th></th>
<th>Unity</th>
<th>Godot</th>
</tr>
</thead>
<tbody>
<tr>
<td>License</td>
<td>Proprietary, closed, free license with revenue caps and usage restrictions</td>
<td>MIT License, free and fully open source without any restriction</td>
</tr>
<tr>
<td>OS (editor)</td>
<td>Windows, OSX, Linux (unofficial and unsupported)</td>
<td>Windows, X11 (Linux, *BSD), Haiku, OSX</td>
</tr>
</tbody>
</table>
| OS (export)    | Desktop: Windows, Linux/SteamOS, OSX  
Mobile: Android, iOS, Windows Phone, Tizen,  
Web: WebGL  
Consoles: PS4, PS Vita, XBox One, XBox 360, WiiU, 3DS  
VR: Oculus Rift, SteamVR, Google Cardboard, Playstation VR, Gear VR, HoloLens  
TV: AndroidTV, Samsung SMARTTV, tvOS | Desktop: Windows, X11, OSX  
Mobile: Android, iOS, Blackberry (deprecated)  
Web: WebGL, HTML5 (via emscripten, broken) |
| Scene system   | Component/Scene (GameObject > Component)  
Prefabs | Scene tree and nodes, allowing scenes to be nested and/or inherit other scenes |
| Third-party tools | Visual Studio or SharpEditor | Android SDK for Android export  
External editors are possible |
| Killer features | Huge community  
Large assets store | Scene System  
Animation Pipeline  
Easy to write Shaders  
Debug on Device |

### 3.2.2 The editor

Godot Engine provides a rich-featured editor that allows you to build your games. The pictures below display both editors with colored blocks to indicate common functionalities.
Note that Godot editor allows you to dock each panel at the side of the scene editor you wish.

While both editors may seem similar, there are many differences below the surface. Both let you organize the project using the filesystem, but Godot approach is simpler, with a single configuration file, minimalist text format, and no metadata. All this contributes to Godot being much friendlier to VCS systems such as Git, Subversion or Mercurial.

Godot’s Scene panel is similar to Unity’s Hierarchy panel but, as each node has a specific function, the approach used by Godot is more visually descriptive. In other words, it’s easier to understand what a specific scene does at a glance.

The Inspector in Godot is more minimalist and designed to only show properties. Thanks to this, objects can export a much larger amount of useful parameters to the user, without having to hide functionality in language APIs. As a plus, Godot allows animating any of those properties visually, so changing colors, textures, enumerations or even links to
resources in real-time is possible without involving code.

Finally, the Toolbar at the top of the screen is similar in the sense that it allows controlling the project playback, but projects in Godot run in a separate window, as they don’t execute inside the editor (but the tree and objects can still be explored in the debugger window).

This approach has the disadvantage that the running game can’t be explored from different angles (though this may be supported in the future, and displaying collision gizmos in the running game is already possible), but in exchange has several advantages:

- Running the project and closing it is very fast (Unity has to save, run the project, close the project and then reload the previous state).
- Live editing is a lot more useful, because changes done to the editor take effect immediately in the game, and are not lost (nor have to be synced) when the game is closed. This allows fantastic workflows, like creating levels while you play them.
- The editor is more stable, because the game runs in a separate process.

Finally, the top toolbar includes a menu for remote debugging. These options make it simple to deploy to a device (connected phone, tablet or browser via HTML5), and debug/live edit on it after the game was exported.

### 3.2.3 The scene system

This is the most important difference between Unity and Godot, and actually the favourite feature of most Godot users. Unity’s scene system consist in embedding all the required assets in a scene, and link them together by setting components and scripts to them. Godot’s scene system is different: it actually consists in a tree made of nodes. Each node serves a purpose: Sprite, Mesh, Light... Basically, this is similar to Unity scene system. However, each node can have multiple children, which make each a subscene of the main scene. This means you can compose a whole scene with different scenes, stored in different files.

For example, think of a platformer level. You would compose it with multiple elements:

- Bricks
- Coins
- The player
- The enemies

In Unity, you would put all the GameObjects in the scene: the player, multiple instances of enemies, bricks everywhere to form the ground of the level, and multiple instances of coins all over the level. You would then add various components to each element to link them and add logic in the level: for example, you’d add a BoxCollider2D to all the elements of the scene so that they can collide. This principle is different in Godot.

In Godot, you would split your whole scene into 3 separate, smaller scenes, which you would then instance in the main scene.

1. First, a scene for the Player alone.

Consider the player as a reusable element in other levels. It is composed of one node in particular: an AnimatedSprite node, which contains the sprite textures to form various animations (for example, walking animation)

2. Second, a scene for the Enemy.

There again, an enemy is a reusable element in other levels. It is almost the same as the Player node - the only differences are the script (that manages AI, mostly) and sprite textures used by the AnimatedSprite.

3. Lastly, the Level scene.
Godot Engine Documentation, Release 2.1

It is composed of Bricks (for platforms), Coins (for the player to grab) and a certain number of instances of the previous Enemy scene. These will be different, separate enemies, whose behaviour and appearance will be the same as defined in the Enemy scene. Each instance is then considered as a node in the Level scene tree. Of course, you can set different properties for each enemy node (to change its color for example).

Finally, the main scene would then be composed of one root node with 2 children: a Player instance node, and a Level instance node. The root node can be anything, generally a “root” type such as “Node” which is the most global type, or “Node2D” (root type of all 2D-related nodes), “Spatial” (root type of all 3D-related nodes) or “Control” (root type of all GUI-related nodes).

As you can see, every scene is organized as a tree. The same goes for nodes’ properties: you don’t add a collision component to a node to make it collidable like Unity does. Instead, you make this node a child of a new specific node that has collision properties. Godot features various collision types nodes, depending of the use (see the Physics introduction).

- Question: What are the advantages of this system? Wouldn’t this system potentially increase the depth of the scene tree? Besides, Unity allows organizing GameObjects by putting them in empty GameObjects.

  - First, this system is closer to the well-known Object-Oriented paradigm: Godot provides a number of nodes which are not clearly “Game Objects”, but they provide their children with their own capabilities: this is inheritance.

  - Second, it allows the extraction a subtree of scene to make it a scene of its own, which answers to the second and third questions: even if a scene tree gets too deep, it can be split into smaller subtrees. This also allows a better solution for reusability, as you can include any subtree as a child of any node. Putting multiple nodes in an empty GameObject in Unity does not provide the same possibility, apart from a visual organization.

These are the most important concepts you need to remember: “node”, “parent node” and “child node”.

70 Chapter 3. Editor manual
We previously observed that there is no perfect solution to set a project architecture. Any solution will work for Unity and Godot, so this point has a lesser importance.

However, we often observe a common architecture for Unity projects, which consists in having one Assets folder in
the root directory, that contains various folders, one per type of asset: Audio, Graphics, Models, Materials, Scripts, Scenes, etc.

As described before, Godot scene system allows splitting scenes in smaller scenes. Since each scene and subscene is actually one scene file in the project, we recommend organizing your project a bit differently. This wiki provides a page for this: Project organization.

### 3.2.5 Where are my prefabs?

The concept of prefabs as provided by Unity is a ‘template’ element of the scene. It is reusable, and each instance of the prefab that exists in the scene has an existence of its own, but all of them have the same properties as defined by the prefab.

Godot does not provide prefabs as such, but this functionality is here again filled thanks to its scene system: as we saw the scene system is organized as a tree. Godot allows you to save a subtree of a scene as its own scene, thus saved in its own file. This new scene can then be instanced as many times as you want. Any change you make to this new, separate scene will be applied to its instances. However, any change you make to the instance will not have any impact on the ‘template’ scene.

![Diagram of Godot scene system]

To be precise, you can modify the parameters of the instance in the Inspector panel. However, the nodes that compose this instance are locked and you can unlock them if you need to by clicking the clapperboard icon next to the instance in the Scene tree, and select “Editable children” in the menu. You don’t need to do this to add new children nodes to this node, but remember that these new children will belong to the instance, not the ‘template’ scene. If you want to
add new children to all the instances of your ‘template’ scene, then you need to add it once in the ‘template’ scene.

3.2.6 Glossary correspondance

GameObject -> Node Add a component -> Inheriting Prefab -> Externalized branch

3.2.7 Scripting : From C# to GDScript

Design

As you may know already, Unity supports 2 scripting languages for its API: C# and Javascript (called UnityScript). Both languages can be used in the same project (but not in the same file, of course). Choosing one instead of the other is a matter of personal taste, as performances seem not to be affected that much by the use of Javascript as long as the project remains small. C# benefits from its integration with Visual Studio and other specific features, such as static typing.

Godot provides its own scripting language: GDScript. This language borrows its syntax to Python, but is not related to it. If you wonder about why GDScript and not Lua, C# or any other, please read GDScript and FAQ pages. GDScript is strongly attached to Godot API, but it is really easy to learn: between 1 evening for an experimented programmer and 1 week for a complete beginner.

Unity allows you to attach as many scripts as you want to a GameObject. Each script adds a behaviour to the GameObject: for example, you can attach a script so that it reacts to the player’s controls, and another that controls its specific game logic.

In Godot, you can only attach one script per node. You can use either an external GDScript file, or include it directly in the node. If you need to attach more scripts to one node, then you may consider 2 solutions, depending on your scene and on what you want to achieve:

- either add a new node between your target node and its current parent, then add a script to this new node.
- or, you can split your target node into multiple children and attach one script to each of them.
As you can see, it can be easy to turn a scene tree to a mess. This is why it is important to have a real reflection, and consider splitting a complicated scene into multiple, smaller branches.

**Connections : groups and signals**

You can control nodes by accessing them using a script, and call functions (built-in or user-defined) on them. But there’s more: you can also place them in a group and call a function on all nodes contained in this group! This is explained in this page.

But there’s more! Certain nodes throw signals when certain actions happen. You can connect these signals to call a specific function when they happen. Note that you can define your own signals and send them whenever you want. This feature is documented here.

**3.2.8 Using Godot in C++**

Just for your information, Godot also allows you to develop your project directly in C++ by using its API, which is not possible with Unity at the moment. As an example, you can consider Godot Engine’s editor as a “game” written in C++ using Godot API.

If you are interested in using Godot in C++, you may want to start reading the Developing in C++ page.

**3.3 Command line tutorial**

Some developers like using the command line extensively. Godot is designed to be friendly to them, so here are the steps for working entirely from the command line. Given the engine relies on little to no external libraries, initialization times are pretty fast, making it suitable for this workflow.

**3.3.1 Path**

It is recommended that your godot binary is in your PATH environment variable, so it can be executed easily from any place by typing `godot`. You can do so on Linux by placing the Godot binary in `/usr/local/bin` and making sure it is called `godot`.

**3.3.2 Creating a project**

Creating a project from the command line is simple, just navigate the shell to the desired place and just make an engine.cfg file exist, even if empty.

```
user@host:~$ mkdir newgame
user@host:~$ cd newgame
user@host:/newgame$ touch engine.cfg
```

That alone makes for an empty Godot project.

**3.3.3 Running the editor**

Running the editor is done by executing godot with the `-e` flag. This must be done from within the project directory, or a subdirectory, otherwise the command is ignored and the project manager appears.
If a scene has been created and saved, it can be edited later by running the same code with that scene as argument.

```
user@host:~/newgame$ godot -e scene.xml
```

### 3.3.4 Erasing a scene

Godot is friends with your filesystem, and will not create extra metadata files, simply use `rm` to erase a file. Make sure nothing references that scene, or else an error will be thrown upon opening.

```
user@host:~/newgame$ rm scene.xml
```

### 3.3.5 Running the game

To run the game, simply execute Godot within the project directory or subdirectory.

```
user@host:~/newgame$ godot
```

When a specific scene needs to be tested, pass that scene to the command line.

```
user@host:~/newgame$ godot scene.xml
```

### 3.3.6 Debugging

Catching errors in the command line can be a difficult task because they just fly by. For this, a command line debugger is provided by adding `-d`. It works for both running the game or a simple scene.

```
user@host:~/newgame$ godot -d
```

```
user@host:~/newgame$ godot -d scene.xml
```

### 3.3.7 Exporting

Exporting the project from the command line is also supported. This is specially useful for continuous integration setups. The version of Godot that is headless (server build, no video) is ideal for this.

```
user@host:~/newgame$ godot -export "Linux X11" /var/builds/project
user@host:~/newgame$ godot -export Android /var/builds/project.apk
```

The platform names recognized by the `-export` switch are the same as displayed in the export wizard of the editor. To get a list of supported platforms from the command line, just try exporting to a non-recognized platform and the full listing of platforms your configuration supports will be shown.

To export a debug version of the game, use the `-export_debug` switch instead of `-export`. Their parameters and usage are the same.
It is possible to run a simple .gd script from the command line. This feature is specially useful in very large projects, for batch conversion of assets or custom import/export.

The script must inherit from SceneTree or MainLoop.

Here is a simple example of how it works:

```
#sayhello.gd
extends SceneTree

func _init():
    print("Hello!")
    quit()
```

And how to run it:

```
user@host:~/newgame$ godot -s sayhello.gd
Hello!
```

If no engine.cfg exists at the path, current path is assumed to be the current working directory (unless --path is specified).
4.1 Math

4.1.1 Vector math

Introduction

This small tutorial aims to be a short and practical introduction to vector math, useful for 3D but also 2D games. Again, vector math is not only useful for 3D but also 2D games. It is an amazing tool once you get the grasp of it and makes programming of complex behaviors much simpler.

It often happens that young programmers rely too much on the incorrect math for solving a wide array of problems, for example using only trigonometry instead of vector math for 2D games.

This tutorial will focus on practical usage, with immediate application to the art of game programming.

Coordinate systems (2D)

Typically, we define coordinates as an (x,y) pair, x representing the horizontal offset and y the vertical one. This makes sense given the screen is just a rectangle in two dimensions. As an example, here is a position in 2D space:

A position can be anywhere in space. The position (0,0) has a name, it’s called the origin. Remember this term well because it has more implicit uses later. The (0,0) of a n-dimensions coordinate system is the origin.

In vector math, coordinates have two different uses, both equally important. They are used to represent a position but also a vector. The same position as before, when imagined as a vector, has a different meaning.
When imagined as a vector, two properties can be inferred, the **direction** and the **magnitude**. Every position in space can be a vector, with the exception of the **origin**. This is because coordinates (0,0) can’t represent direction (magnitude 0).

**Direction**

Direction is simply towards where the vector points to. Imagine an arrow that starts at the **origin** and goes towards a position. The tip of the arrow is in the position, so it always points outwards, away from the origin. Imagining vectors as arrows helps a lot.
Magnitude

Finally, the length of the vector is the distance from the origin to the position. Obtaining the length from a vector is easy, just use the Pythagorean Theorem.

```javascript
var len = sqrt( x*x + y*y )
```

But... angles?

But why not using an angle? After all, we could also think of a vector as an angle and a magnitude, instead of a direction and a magnitude. Angles also are a more familiar concept.

To say truth, angles are not that useful in vector math, and most of the time they are not dealt with directly. Maybe they work in 2D, but in 3D a lot of what can usually be done with angles does not work anymore.

Still, using angles is still not an excuse, even for 2D. Most of what takes a lot of work with angles in 2D, is still much more natural easier to accomplish with vector math. In vector math, angles are useful only as measure, but take little part in the math. So, give up the trigonometry already, prepare to embrace vectors!

In any case, obtaining an angle from a vector is easy and can be accomplished with trig... er, what was that? I mean, the \texttt{atan2()} function.

Vectors in Godot

To make examples easier, it is worth explaining how vectors are implemented in GDScript. GDscript has both \texttt{Vector2} and \texttt{Vector3}, for 2D and 3D math respectively. Godot uses Vector classes as both position and direction. They also contain x and y (for 2D) and x, y and z (for 3D) member variables.

```javascript
# create a vector with coordinates (2,5)
var a = Vector2(2,5)
# create a vector and assign x and y manually
var b = Vector2()
b.x = 7
b.y = 8
```

When operating with vectors, it is not necessary to operate on the members directly (in fact this is much slower). Vectors support regular arithmetic operations:

```javascript
# add a and b
var c = a + b
# will result in c vector, with value (9,13)
```

It is the same as doing:

```javascript
var c = Vector2()
c.x = a.x + b.x
c.y = a.y + b.y
```

Except the former is way more efficient and readable.

Regular arithmetic operations such as addition, subtraction, multiplication and division are supported.

Vector multiplication and division can also be mixed with single-digit numbers, also named \texttt{scalars}.
Godot Engine Documentation, Release 2.1

```godot
# multiplication of vector by scalar
var c = a*2.0
# will result in c vector, with value (4,10)
```

Which is the same as doing

```godot
var c = Vector2()
c.x = a.x*2.0
c.y = a.y*2.0
```

Except, again, the former is way more efficient and readable.

**Perpendicular vectors**

Rotating a 2D vector 90° degrees to either side, left or right, is really easy, just swap x and y, then negate either x or y (direction of rotation depends on which is negated).

```
Left
(-4,2)

(2,4)

Right
(4,-2)
```

Example:

```godot
var v = Vector2(0,1)

# rotate right (clockwise)
var v_right = Vector2(v.y, -v.x)

# rotate left (counter-clockwise)
var v_left = Vector2(-v.y, v.x)
```

This is a handy trick that is often of use. It is impossible to do with 3D vectors, because there are an infinite amount of perpendicular vectors.

**Unit vectors**

Ok, so we know what a vector is. It has a direction and a magnitude. We also know how to use them in Godot. The next step is learning about **unit vectors**. Any vector with magnitude of length 1 is considered a **unit vector**. In 2D, imagine drawing a circle of radius one. That circle contains all unit vectors in existence for 2 dimensions:
So, what is so special about unit vectors? Unit vectors are amazing. In other words, unit vectors have several, very useful properties.

Can’t wait to know more about the fantastic properties of unit vectors, but one step at a time. So, how is a unit vector created from a regular vector?

### Normalization

Taking any vector and reducing its magnitude to 1.0 while keeping its direction is called normalization. Normalization is performed by dividing the x and y (and z in 3D) components of a vector by its magnitude:

```javascript
var a = Vector2(2, 4)
var m = sqrt(a.x*a.x + a.y*a.y)
a.x /= m
a.y /= m
```

As you might have guessed, if the vector has magnitude 0 (meaning, it’s not a vector but the origin also called null vector), a division by zero occurs and the universe goes through a second big bang, except in reverse polarity and then back. As a result, humanity is safe but Godot will print an error. Remember! Vector(0,0) can’t be normalized!

Of course, Vector2 and Vector3 already provide a method to do this:

```javascript
a = a.normalized()
```

### Dot product

OK, the dot product is the most important part of vector math. Without the dot product, Quake would have never been made. This is the most important section of the tutorial, so make sure to grasp it properly. Most people trying to understand vector math give up here because, despite how simple it is, they can’t make head or tails from it. Why? Here’s why, it’s because...

The dot product takes two vectors and returns a scalar:

```javascript
var s = a.x*b.x + a.y*b.y
```

Yes, pretty much that. Multiply x from vector a by x from vector b. Do the same with y and add it together. In 3D it's pretty much the same:

```javascript
var s = a.x*b.x + a.y*b.y + a.z*b.z
```
I know, it’s totally meaningless! You can even do it with a built-in function:

```javascript
var s = a.dot(b)
```

The order of two vectors does not matter, `a.dot(b)` returns the same value as `b.dot(a)`.

This is where despair begins and books and tutorials show you this formula:

$$\mathbf{A} \cdot \mathbf{B} = \|A\| \|B\| \cos \theta,$$

And you realize it’s time to give up making 3D games or complex 2D games. How can something so simple be so complex? Someone else will have to make the next Zelda or Call of Duty. Top down RPGs don’t look so bad after all. Yeah I hear someone did pretty will with one of those on Steam...

So this is your moment, this is your time to shine. **DO NOT GIVE UP**! At this point, this tutorial will take a sharp turn and focus on what makes the dot product useful. This is, **why** it is useful. We will focus one by one in the use cases for the dot product, with real-life applications. No more formulas that don’t make any sense. Formulas will make sense once you learn what they are useful for.

### Siding

The first useful and most important property of the dot product is to check what side stuff is looking at. Let’s imagine we have any two vectors, `a` and `b`. Any **direction** or **magnitude** (neither **origin**). Does not matter what they are, but let’s imagine we compute the dot product between them.

```javascript
var s = a.dot(b)
```

The operation will return a single floating point number (but since we are in vector world, we call them **scalar**, will keep using that term from now on). This number will tell us the following:

- If the number is greater than zero, both are looking towards the same direction (the angle between them is < 90° degrees).
- If the number is less than zero, both are looking towards opposite direction (the angle between them is > 90° degrees).
- If the number is zero, vectors are shaped in L (the angle between them is 90° degrees).

So let’s think of a real use-case scenario. Imagine Snake is going through a forest, and then there is an enemy nearby. How can we quickly tell if the enemy has seen discovered Snake? In order to discover him, the enemy must be able to see Snake. Let’s say, then that:

- Snake is in position `A`.
- The enemy is in position `B`.
- The enemy is **facing** towards direction vector `F`.

- `a.dot(b)>0`
- `a.dot(b)<0`
- `a.dot(b)==0`
So, let’s create a new vector $\mathbf{BA}$ that goes from the guard ($\mathbf{B}$) to Snake ($\mathbf{A}$), by subtracting the two:

```python
var BA = A - B
```

Ideally, if the guard was looking straight towards snake, to make eye to eye contact, it would do it in the same direction as vector $\mathbf{BA}$.

If the dot product between $\mathbf{F}$ and $\mathbf{BA}$ is greater than 0, then Snake will be discovered. This happens because we will be able to tell that the guard is facing towards him:

```python
if (BA.dot(F) > 0):
    print("!")
```

Seems Snake is safe so far.

**Siding with unit vectors**

Ok, so now we know that dot product between two vectors will let us know if they are looking towards the same side, opposite sides or are just perpendicular to each other.
This works the same with all vectors, no matter the magnitude so unit vectors are not the exception. However, using the same property with unit vectors yields an even more interesting result, as an extra property is added:

- If both vectors are facing towards the exact same direction (parallel to each other, angle between them is 0°), the resulting scalar is 1.
- If both vectors are facing towards the exact opposite direction (parallel to each other, but angle between them is 180°), the resulting scalar is -1.

This means that dot product between unit vectors is always between the range of 1 and -1. So Again...

- If their angle is 0° dot product is 1.
- If their angle is 90°, then dot product is 0.
- If their angle is 180°, then dot product is -1.

Uh.. this is oddly familiar... seen this before... where?

Let's take two unit vectors. The first one is pointing up, the second too but we will rotate it all the way from up (0°) to down (180° degrees)...

![Diagram showing rotation from 0° to 180°](image)

While plotting the resulting scalar!

![Graph showing cosine function](image)

Aha! It all makes sense now, this is a Cosine function!

We can say that, then, as a rule...

The dot product between two unit vectors is the cosine of the angle between those two vectors. So, to obtain the angle between two vectors, we must do:
var angle_in_radians = acos(a.dot(b))

What is this useful for? Well obtaining the angle directly is probably not as useful, but just being able to tell the angle is useful for reference. One example is in the Kinematic Character demo, when the character moves in a certain direction then we hit an object. How to tell if what we hit is the floor?

By comparing the normal of the collision point with a previously computed angle.

The beauty of this is that the same code works exactly the same and without modification in 3D. Vector math is, in a great deal, dimension-amount-independent, so adding or removing an axis only adds very little complexity.

**Planes**

The dot product has another interesting property with unit vectors. Imagine that perpendicular to that vector (and through the origin) passes a plane. Planes divide the entire space into positive (over the plane) and negative (under the plane), and (contrary to popular belief) you can also use their math in 2D:

![Diagram of planes](image)

Unit vectors that are perpendicular to a surface (so, they describe the orientation of the surface) are called *unit normal vectors*. Though, usually they are just abbreviated as *normals*. Normals appear in planes, 3D geometry (to determine where each face or vertex is siding), etc. A normal is a *unit vector*, but it’s called normal because of its usage. (Just like we call Origin to (0,0)!).

It’s as simple as it looks. The plane passes by the origin and the surface of it is perpendicular to the unit vector (or normal). The side towards the vector points to is the positive half-space, while the other side is the negative half-space. In 3D this is exactly the same, except that the plane is an infinite surface (imagine an infinite, flat sheet of paper that you can orient and is pinned to the origin) instead of a line.

**Distance to plane**

Now that it’s clear what a plane is, let’s go back to the dot product. The dot product between a *unit vector* and any *point in space* (yes, this time we do dot product between vector and position), returns the *distance from the point to the plane*:

```javascript
var distance = normal.dot(point)
```

But not just the absolute distance, if the point is in the negative half space the distance will be negative, too:
This allows us to tell which side of the plane a point is.

**Away from the origin**

I know what you are thinking! So far this is nice, but real planes are everywhere in space, not only passing through the origin. You want real plane action and you want it now.

Remember that planes not only split space in two, but they also have polarity. This means that it is possible to have perfectly overlapping planes, but their negative and positive half-spaces are swapped.

With this in mind, let’s describe a full plane as a normal $N$ and a distance from the origin scalar $D$. Thus, our plane is represented by $N$ and $D$. For example:

For 3D math, Godot provides a `Plane` built-in type that handles this.

Basically, $N$ and $D$ can represent any plane in space, be it for 2D or 3D (depending on the amount of dimensions of $N$) and the math is the same for both. It’s the same as before, but $D$ is the distance from the origin to the plane, travelling in $N$ direction. As an example, imagine you want to reach a point in the plane, you will just do:

```javascript
var point_in_plane = N*D
```

This will stretch (resize) the normal vector and make it touch the plane. This math might seem confusing, but it’s actually much simpler than it seems. If we want to tell, again, the distance from the point to the plane, we do the same but adjusting for distance:

```javascript
var distance = N.dot(point) - D
```
The same thing, using a built-in function:

```javascript
var distance = plane.distance_to(point)
```

This will, again, return either a positive or negative distance.

Flipping the polarity of the plane is also very simple, just negate both N and D. This will result in a plane in the same position, but with inverted negative and positive half spaces:

```javascript
N = -N
D = -D
```

Of course, Godot also implements this operator in `Plane`, so doing:

```javascript
var inverted_plane = -plane
```

Will work as expected.

So, remember, a plane is just that and its main practical use is calculating the distance to it. So, why is it useful to calculate the distance from a point to a plane? It’s extremely useful! Let’s see some simple examples..

### Constructing a plane in 2D

Planes clearly don’t come out of nowhere, so they must be built. Constructing them in 2D is easy, this can be done from either a normal (unit vector) and a point, or from two points in space.

In the case of a normal and a point, most of the work is done, as the normal is already computed, so just calculate D from the dot product of the normal and the point:

```javascript
var N = normal
var D = normal.dot(point)
```

For two points in space, there are actually two planes that pass through them, sharing the same space but with normal pointing to the opposite directions. To compute the normal from the two points, the direction vector must be obtained first, and then it needs to be rotated 90° degrees to either side:

```javascript
# calculate vector from a to b
var dvec = (point_b - point_a).normalized()
# rotate 90 degrees
var normal = Vector2(dvec.y, -dvec.x)
# or alternatively
# var normal = Vector2(-dvec.y, dvec.x)
# depending the desired side of the normal
```

The rest is the same as the previous example, either point_a or point_b will work since they are in the same plane:

```javascript
var N = normal
var D = normal.dot(point_a)
# this works the same
# var D = normal.dot(point_b)
```

Doing the same in 3D is a little more complex and will be explained further down.

### Some examples of planes

Here is a simple example of what planes are useful for. Imagine you have a convex polygon. For example, a rectangle, a trapezoid, a triangle, or just any polygon where faces that don’t bend inwards.
For every segment of the polygon, we compute the plane that passes by that segment. Once we have the list of planes, we can do neat things, for example checking if a point is inside the polygon.

We go through all planes, if we can find a plane where the distance to the point is positive, then the point is outside the polygon. If we can’t, then the point is inside.

Code should be something like this:

```python
var inside = true
for p in planes:
    # check if distance to plane is positive
    if (N.dot(point) - D > 0):
        inside = false
        break # with one that fails, it’s enough
```

Pretty cool, huh? But this gets much better! With a little more effort, similar logic will let us know when two convex polygons are overlapping too. This is called the Separating Axis Theorem (or SAT) and most physics engines use this to detect collision.

The idea is really simple! With a point, just checking if a plane returns a positive distance is enough to tell if the point is outside. With another polygon, we must find a plane where all the **other** polygon points return a positive distance to it. This check is performed with the planes of A against the points of B, and then with the planes of B against the points of A:
Code should be something like this:

```python
var overlapping = true

for p in planes_of_A:
    var all_out = true
    for v in points_of_B:
        if p.distance_to(v) < 0:
            all_out = false
            break

if (all_out):
    # a separating plane was found
    # do not continue testing
    overlapping = false
    break

if (overlapping):
    # only do this check if no separating plane
    # was found in planes of A
    for p in planes_of_B:
```

4.1. Math
```python
var all_out = true
for v in points_of_A:
    if (p.distance_to(v) < 0):
        all_out = false
        break

if (all_out):
    overlapping = false
    break

if (overlapping):
    print("Polygons Collided!")
```

As you can see, planes are quite useful, and this is the tip of the iceberg. You might be wondering what happens with non convex polygons. This is usually just handled by splitting the concave polygon into smaller convex polygons, or using a technique such as BSP (which is not used much nowadays).

**Cross product**

Quite a lot can be done with the dot product! But the party would not be complete without the cross product. Remember back at the beginning of this tutorial? Specifically how to obtain a perpendicular (rotated 90 degrees) vector by swapping x and y, then negating either of them for right (clockwise) or left (counter-clockwise) rotation? That ended up being useful for calculating a 2D plane normal from two points.

As mentioned before, no such thing exists in 3D because a 3D vector has infinite perpendicular vectors. It would also not make sense to obtain a 3D plane from 2 points, as 3 points are needed instead.

To aid in this kind stuff, the brightest minds of humanity’s top mathematicians brought us the **cross product**.

The cross product takes two vectors and returns another vector. The returned third vector is always perpendicular to the first two. The source vectors, of course, must not be the same, and must not be parallel or opposite, else the resulting vector will be (0,0,0):
The formula for the cross product is:

```javascript
var c = Vector3();
c.x = (a.y * b.z) - (a.z * b.y);
c.y = (a.z * b.x) - (a.x * b.z);
c.z = (a.x * b.y) - (a.y * b.x);
```

This can be simplified, in Godot, to:

```javascript
var c = a.cross(b);
```

However, unlike the dot product, doing `a.cross(b)` and `b.cross(a)` will yield different results. Specifically, the returned vector will be negated in the second case. As you might have realized, this coincides with creating perpendicular vectors in 2D. In 3D, there are also two possible perpendicular vectors to a pair of 2D vectors.

Also, the resulting cross product of two unit vectors is not a unit vector. Result will need to be renormalized.

**Area of a triangle**

Cross product can be used to obtain the surface area of a triangle in 3D. Given a triangle consisting of 3 points, A, B and C:
Take any of them as a pivot and compute the adjacent vectors to the other two points. As example, we will use B as a pivot:

```plaintext
var BA = A - B
var BC = C - B
```
Compute the cross product between $\mathbf{BA}$ and $\mathbf{BC}$ to obtain the perpendicular vector $\mathbf{P}$:

```javascript
var P = BA.cross(BC)
```
The length (magnitude) of \( \mathbf{P} \) is the surface area of the parallelogram built by the two vectors \( \mathbf{BA} \) and \( \mathbf{BC} \), therefore the surface area of the triangle is half of it.

\[
\text{var} \quad \text{area} = \frac{\text{\mathbf{P}.length()}}{2}
\]

**Plane of the triangle**

With \( \mathbf{P} \) computed from the previous step, normalize it to get the normal of the plane.

\[
\text{var} \quad \mathbf{N} = \frac{\mathbf{P}}{\mathbf{P}.length()}
\]

And obtain the distance by doing the dot product of \( \mathbf{P} \) with any of the 3 points of the \( \mathbf{ABC} \) triangle:

\[
\text{var} \quad \mathbf{D} = \mathbf{P} \cdot \mathbf{A}
\]

Fantastic! You computed the plane from a triangle!

Here’s some useful info (that you can find in Godot source code anyway). Computing a plane from a triangle can result in 2 planes, so a sort of convention needs to be set. This usually depends (in video games and 3D visualization) to use the front-facing side of the triangle.

In Godot, front-facing triangles are those that, when looking at the camera, are in clockwise order. Triangles that look Counter-clockwise when looking at the camera are not drawn (this helps to draw less, so the back-part of the objects is not drawn).

To make it a little clearer, in the image below, the triangle \( \mathbf{ABC} \) appears clock-wise when looked at from the *Front Camera*, but to the *Rear Camera* it appears counter-clockwise so it will not be drawn.

Normals of triangles often are sided towards the direction they can be viewed from, so in this case, the normal of triangle \( \mathbf{ABC} \) would point towards the front camera:
So, to obtain N, the correct formula is:

```javascript
# clockwise normal from triangle formula
var N = (A-C).cross(A-B).normalized()
# for counter-clockwise:
# var N = (A-B).cross(A-C).normalized()
var D = N.dot(A)
```

**Collision detection in 3D**

This is another bonus bit, a reward for being patient and keeping up with this long tutorial. Here is another piece of wisdom. This might not be something with a direct use case (Godot already does collision detection pretty well) but it’s a really cool algorithm to understand anyway, because it’s used by almost all physics engines and collision detection libraries :)

Remember that converting a convex shape in 2D to an array of 2D planes was useful for collision detection? You could detect if a point was inside any convex shape, or if two 2D convex shapes were overlapping.

Well, this works in 3D too, if two 3D polyhedral shapes are colliding, you won’t be able to find a separating plane. If a separating plane is found, then the shapes are definitely not colliding.

To refresh a bit a separating plane means that all vertices of polygon A are in one side of the plane, and all vertices of polygon B are in the other side. This plane is always one of the face-planes of either polygon A or polygon B.

In 3D though, there is a problem to this approach, because it is possible that, in some cases a separating plane can’t be found. This is an example of such situation:
To avoid it, some extra planes need to be tested as separators, these planes are the cross product between the edges of polygon A and the edges of polygon B.
So the final algorithm is something like:

```javascript
var overlapping = true

for p in planes_of_A:
    var all_out = true
    for v in points_of_B:
        if (p.distance_to(v) < 0):
            all_out = false
            break

    if (all_out):
        # a separating plane was found
        # do not continue testing
        overlapping = false
        break

if (overlapping):
    # only do this check if no separating plane
    # was found in planes of A
```

4.1. Math
for p in planes_of_B:
    var all_out = true
for v in points_of_A:
    if (p.distance_to(v) < 0):
        all_out = false
        break
    if (all_out):
        overlapping = false
        break
if (overlapping):
    for ea in edges_of_A:
        for eb in edges_of_B:
            var n = ea.cross(eb)
            if (n.length() == 0):
                continue

            var max_A = -1e20 # tiny number
            var min_A = 1e20 # huge number

            # we are using the dot product directly
            # so we can map a maximum and minimum range
            # for each polygon, then check if they
            # overlap.

            for v in points_of_A:
                var d = n.dot(v)
                if (d > max_A):
                    max_A = d
                if (d < min_A):
                    min_A = d

            var max_B = -1e20 # tiny number
            var min_B = 1e20 # huge number

            for v in points_of_B:
                var d = n.dot(v)
                if (d > max_B):
                    max_B = d
                if (d < min_B):
                    min_B = d

            if (min_A > max_B or min_B > max_A):
                # not overlapping!
                overlapping = false
                break

            if (not overlapping):
                break

if (overlapping):
    print("Polygons collided!")

This was all! Hope it was helpful, and please give feedback and let know if something in this tutorial is not clear! You should be now ready for the next challenge... Matrices and transforms!
4.1.2 Matrices and transforms

Introduction

Before reading this tutorial, it is advised to read the previous one about Vector math as this one is a direct continuation. This tutorial will be about transformations and will cover a little about matrices (but not in-depth).

Transformations are most of the time applied as translation, rotation and scale so they will be considered as priority here.

Oriented coordinate system (OCS)

Imagine we have a spaceship somewhere in space. In Godot this is easy, just move the ship somewhere and rotate it:

![Spaceship diagram](image)

Ok, so in 2D this looks simple, a position and an angle for a rotation. But remember, we are grown ups here and don’t use angles (plus, angles are not really even that useful when working in 3D).

We should realize that at some point, someone designed this spaceship. Be it for 2D in a drawing such as Paint.net, Gimp, Photoshop, etc. or in 3D through a 3D DCC tool such as Blender, Max, Maya, etc.

When it was designed, it was not rotated. It was designed in its own coordinate system.
This means that the tip of the ship has a coordinate, the fin has another, etc. Be it in pixels (2D) or vertices (3D).

So, let’s recall again that the ship was somewhere in space:

How did it get there? What moved it and rotated it from the place it was designed to its current position? The answer is... a **transform**, the ship was transformed from their original position to the new one. This allows the ship to be displayed where it is.

But transform is too generic of a term to describe this process. To solve this puzzle, we will superimpose the ship’s original design position at their current position:
So, we can see that the “design space” has been transformed too. How can we best represent this transformation? Let’s use 3 vectors for this (in 2D), a unit vector pointing towards X positive, a unit vector pointing towards Y positive and a translation.

Let’s call the 3 vectors “X”, “Y” and “Origin”, and let’s also superimpose them over the ship so it makes more sense:
Ok, this is nicer, but it still does not make sense. What do X,Y and Origin have to do with how the ship got there? Well, let’s take the point from top tip of the ship as reference:

And let’s apply the following operation to it (and to all the points in the ship too, but we’ll track the top tip as our reference point):

```var new_pos = pos - origin
```

Doing this to the selected point will move it back to the center:
This was expected, but then let’s do something more interesting. Use the dot product of X and the point, and add it to the dot product of Y and the point:

```javascript
var final_pos = x.dot(new_pos) + y.dot(new_pos)
```

Then what we have is.. wait a minute, it’s the ship in its design position!

How did this black magic happen? The ship was lost in space, and now it’s back home!

It might seem strange, but it does have plenty of logic. Remember, as we have seen in the Vector math, what happened is that the distance to X axis, and the distance to Y axis were computed. Calculating distance in a direction or plane was one of the uses for the dot product. This was enough to obtain back the design coordinates for every point in the ship.

So, what he have been working with so far (with X, Y and Origin) is an Oriented Coordinate System*. X an Y are the **Basis*, and *Origin* is the offset.
Basis

We know what the Origin is. It’s where the 0,0 (origin) of the design coordinate system ended up after being transformed to a new position. This is why it’s called Origin, But in practice, it’s just an offset to the new position.

The Basis is more interesting. The basis is the direction of X and Y in the OCS from the new, transformed location. It tells what has changed, in either 2D or 3D. The Origin (offset) and Basis (direction) communicate “Hey, the original X and Y axes of your design are right here, pointing towards these directions.”

So, let’s change the representation of the basis. Instead of 2 vectors, let’s use a matrix.

\[
M = \begin{bmatrix}
X_x & X_y \\
Y_x & Y_y
\end{bmatrix}
\]

The vectors are up there in the matrix, horizontally. The next problem now is that.. what is this matrix thing? Well, we’ll assume you’ve never heard of a matrix.

Transforms in Godot

This tutorial will not explain matrix math (and their operations) in depth, only its practical use. There is plenty of material for that, which should be a lot simpler to understand after completing this tutorial. We’ll just explain how to use transforms.

Matrix32

Matrix32 is a 3x2 matrix. It has 3 Vector2 elements and it’s used for 2D. The “X” axis is the element 0, “Y” axis is the element 1 and “Origin” is element 2. It’s not divided in basis/origin for convenience, due to its simplicity.

```javascript
var m = Matrix32();
var x = m[0] # 'X'
var y = m[1] # 'Y'
var o = m[2] # 'Origin'
```

Most operations will be explained with this datatype (Matrix32), but the same logic applies to 3D.

Identity

By default, Matrix32 is created as an “identity” matrix. This means:

- ‘X’ Points right: Vector2(1,0)
- ‘Y’ Points up (or down in pixels): Vector2(0,1)
- ‘Origin’ is the origin Vector2(0,0)
It’s easy to guess that an identity matrix is just a matrix that aligns the transform to its parent coordinate system. It’s an OCS that hasn’t been translated, rotated or scaled. All transform types in Godot are created with identity.

**Operations**

**Rotation**

Rotating Matrix32 is done by using the “rotated” function:

```javascript
var m = Matrix32()
m = m.rotated(PI/2) // rotate 90°
```

**Translation**

There are two ways to translate a Matrix32, the first one is just moving the origin:

```javascript
# Move 2 units to the right
var m = Matrix32()
```
This will always work in global coordinates.

If instead, translation is desired in *local* coordinates of the matrix (towards where the *basis* is oriented), there is the `Matrix32.translated()` method:

```
# Move 2 units towards where the basis is oriented
var m = Matrix32()
  m = m.rotated(PI/2) # rotate 90°
  m = m.translated( Vector2(2,0) )
```

**Scale**

A matrix can be scaled too. Scaling will multiply the basis vectors by a vector (X vector by x component of the scale, Y vector by y component of the scale). It will leave the origin alone:
# Make the basis twice its size.

```javascript
var m = Matrix32()
m = m.scaled( Vector2(2,2) )
```

These kind of operations in matrices are accumulative. It means every one starts relative to the previous one. For those who have been living on this planet long enough, a good reference of how transform works is this:
A matrix is used similarly to a turtle. The turtle most likely had a matrix inside (and you are likely learning this many years after discovering Santa is not real).

**Transform**

Transform is the act of switching between coordinate systems. To convert a position (either 2D or 3D) from “designer” coordinate system to the OCS, the “xform” method is used.

```javascript
var new_pos = m.xform(pos)
```

And only for basis (no translation):

```javascript
var new_pos = m.basis_xform(pos)
```

Post - multiplying is also valid:

```javascript
var new_pos = m * pos
```

**Inverse transform**

To do the opposite operation (what we did up there with the rocket), the “xform_inv” method is used:

```javascript
var new_pos = m.xform_inv(pos)
```

Only for Basis:

```javascript
var new_pos = m.basis_xform_inv(pos)
```

Or pre-multiplication:

```javascript
var new_pos = pos * m
```

**Orthonormal matrices**

However, if the Matrix has been scaled (vectors are not unit length), or the basis vectors are not orthogonal (90°), the inverse transform will not work.

In other words, inverse transform is only valid in orthonormal matrices. For this, these cases an affine inverse must be computed.

The transform, or inverse transform of an identity matrix will return the position unchanged:

```javascript
# Does nothing, pos is unchanged
pos = Matrix32().xform(pos)
```

**Affine inverse**

The affine inverse is a matrix that does the inverse operation of another matrix, no matter if the matrix has scale or the axis vectors are not orthogonal. The affine inverse is calculated with the affine_inverse() method:
```javascript
var mi = m.affine_inverse()
var pos = m.xform(pos)
pos = mi.xform(pos)
# pos is unchanged
```

If the matrix is orthonormal, then:

```javascript
# if m is orthonormal, then
pos = mi.xform(pos)
# is the same as
pos = m.xform_inv(pos)
```

### Matrix multiplication

Matrices can be multiplied. Multiplication of two matrices “chains” (concatenates) their transforms. However, as per convention, multiplication takes place in reverse order.

Example:

```javascript
var m = more_transforms * some_transforms
```

To make it a little clearer, this:

```javascript
pos = transform1.xform(pos)
pos = transform2.xform(pos)
```

Is the same as:

```javascript
# note the inverse order
pos = (transform2 * transform1).xform(pos)
```

However, this is not the same:

```javascript
# yields a different results
pos = (transform1 * transform2).xform(pos)
```

Because in matrix math, \( A + B \) is not the same as \( B + A \).

### Multiplication by inverse

Multiplying a matrix by its inverse, results in identity

```javascript
# No matter what A is, B will be identity
B = A.affine_inverse() * A
```

### Multiplication by identity

Multiplying a matrix by identity, will result in the unchanged matrix:

```javascript
# B will be equal to A
B = A * Matrix32()
```
Matrix tips

When using a transform hierarchy, remember that matrix multiplication is reversed! To obtain the global transform for a hierarchy, do:

```godot
var global_xform = parent_matrix * child_matrix
```

For 3 levels:

```godot
# due to reverse order, parenthesis are needed
var global_xform = gradparent_matrix + (parent_matrix + child_matrix)
```

To make a matrix relative to the parent, use the affine inverse (or regular inverse for orthonormal matrices).

```godot
# transform B from a global matrix to one local to A
var B_local_to_A = A.affine_inverse() * B
```

Revert it just like the example above:

```godot
# transform back local B to global B
var B = A * B_local_to_A
```

OK, hopefully this should be enough! Let’s complete the tutorial by moving to 3D matrices.

Matrices & transforms in 3D

As mentioned before, for 3D, we deal with 3 `Vector3` vectors for the rotation matrix, and an extra one for the origin.

Matrix3

Godot has a special type for a 3x3 matrix, named `Matrix3`. It can be used to represent a 3D rotation and scale. Sub vectors can be accessed as:

```godot
var m = Matrix3()
var x = m[0] # Vector3
var y = m[1] # Vector3
var z = m[2] # Vector3
```

Or, alternatively as:

```godot
var m = Matrix3()
var x = m.x # Vector3
var y = m.y # Vector3
var z = m.z # Vector3
```

Matrix3 is also initialized to Identity by default:

\[
\begin{pmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{pmatrix}
\]

\(X\)

\(Y\)

\(Z\)
Rotation in 3D

Rotation in 3D is more complex than in 2D (translation and scale are the same), because rotation is an implicit 2D operation. To rotate in 3D, an axis, must be picked. Rotation, then, happens around this axis.

The axis for the rotation must be a normal vector. As in, a vector that can point to any direction, but length must be one (1.0).

```cpp
#rotate in Y axis
var m3 = Matrix3()
m3 = m3.rotated( Vector3(0,1,0), PI/2 )
```

Transform

To add the final component to the mix, Godot provides the `Transform` type. Transform has two members:

- `basis` (of type `Matrix3`)
- `origin` (of type `Vector3`)

Any 3D transform can be represented with Transform, and the separation of basis and origin makes it easier to work translation and rotation separately.

An example:

```cpp
var t = Transform()
pos = t.xform(pos)  # transform 3D position
pos = t.basis.xform(pos)  # (only rotate)
pos = t.origin + pos  # (only translate)
```

4.2 2D

4.2.1 Viewport and canvas transforms

Introduction

This tutorial is created after a topic that is a little dark for most users, and explains all the 2D transforms going on for nodes from the moment they draw their content locally to the time they are drawn into the screen.

Canvas transform

As mentioned in the previous tutorial, Canvas layers, every CanvasItem node (remember that Node2D and Control based nodes use CanvasItem as their common root) will reside in a Canvas Layer. Every canvas layer has a transform (translation, rotation, scale, etc.) that can be accessed as a `Matrix32`.

Also covered in the previous tutorial, nodes are drawn by default in Layer 0, in the built-in canvas. To put nodes in a different layer, a `CanvasLayer` node can be used.

Global canvas transform

Viewports also have a Global Canvas transform (also a `Matrix32`). This is the master transform and affects all individual Canvas Layer transforms. Generally this transform is not of much use, but is used in the CanvasItem Editor in Godot’s editor.
Stretch transform

Finally, viewports have a *Stretch Transform*, which is used when resizing or stretching the screen. This transform is used internally (as described in *Multiple resolutions*), but can also be manually set on each viewport.

Input events received in the `MainLoop._input_event()` callback are multiplied by this transform, but lack the ones above. To convert InputEvent coordinates to local CanvasItem coordinates, the `CanvasItem.make_input_local()` function was added for convenience.

Transform order

For a coordinate in CanvasItem local properties to become an actual screen coordinate, the following chain of transforms must be applied:
Transform functions

Obtaining each transform can be achieved with the following functions:

<table>
<thead>
<tr>
<th>Type</th>
<th>Transform</th>
</tr>
</thead>
<tbody>
<tr>
<td>CanvasItem</td>
<td>CanvasItem.get_global_transform()</td>
</tr>
<tr>
<td>CanvasLayer</td>
<td>CanvasItem.get_canvas_transform()</td>
</tr>
<tr>
<td>CanvasLayer+GlobalCanvas+Stretch</td>
<td>CanvasItem.get_viewport_transform()</td>
</tr>
</tbody>
</table>

Finally then, to convert a CanvasItem local coordinates to screen coordinates, just multiply in the following order:
Keep in mind, however, that it is generally not desired to work with screen coordinates. The recommended approach is to simply work in Canvas coordinates (CanvasItem.get_global_transform()), to allow automatic screen resolution resizing to work properly.

**Feeding custom input events**

It is often desired to feed custom input events to the scene tree. With the above knowledge, to correctly do this, it must be done the following way:

```javascript
var local_pos = Vector2(10,20) # local to Control/Node2D
var ie = InputEvent()
ie.type = InputEvent.MOUSE_BUTTON
ie.button_index = BUTTON_LEFT
ie.pos = get_viewport_transform() * (get_global_transform() * local_pos)
get_tree().input_event(ie)
```

**4.2.2 Canvas layers**

**Viewport and Canvas items**

Regular 2D nodes, such as Node2D or Control both inherit from CanvasItem, which is the base for all 2D nodes. CanvasItems can be arranged in trees and they will inherit their transform. This means that when moving the parent, the children will be moved too.

These nodes are placed as direct or indirect children to a Viewport, and will be displayed through it.

Viewport has a property “canvas_transform” Viewport.set_canvas_transform(), which allows to transform all the CanvasItem hierarchy by a custom Matrix32 transform. Nodes such as Camera2D, work by changing that transform.

Changing the canvas transform is useful because it is a lot more efficient than moving the root canvas item (and hence the whole scene). Canvas transform is a simple matrix that offsets the whole 2D drawing, so it’s the most efficient way to do scrolling.

**Not enough...**

But this is not enough. There are often situations where the game or application may not want everything transformed by the canvas transform. Examples of this are:

- **Parallax Backgrounds**: Backgrounds that move slower than the rest of the stage.
- **HUD**: Head’s up display, or user interface. If the world moves, the life counter, score, etc. must stay static.
- **Transitions**: Effects used for transitions (fades, blends) may also want it to remain at a fixed location.

How can these problems be solved in a single scene tree?

**CanvasLayers**

The answer is CanvasLayer, which is a node that adds a separate 2D rendering layer for all its children and grandchildren. Viewport children will draw by default at layer “0”, while a CanvasLayer will draw at any numeric layer. Layers with a greater number will be drawn above those with a smaller number. CanvasLayers also have their own
transform, and do not depend of the transform of other layers. This allows the UI to be fixed in-place, while the world
moves.

An example of this is creating a parallax background. This can be done with a CanvasLayer at layer “-1”. The screen
with the points, life counter and pause button can also be created at layer “1”.

Here’s a diagram of how it looks:

![Diagram of scene tree]

CanvasLayers are independent of tree order, and they only depend on their layer number, so they can be instantiated
when needed.

**Performance**

Even though there shouldn’t be any performance limitation, it is not advised to use excessive amount of layers to
arrange drawing order of nodes. The most optimal way will always be arranging them by tree order. 2d nodes also
have a property for controlling their drawing order (see `Node2D.set_z()`).

### 4.2.3 Using tilemaps

**Introduction**

Tilemaps are a simple and quick way to make 2D game levels. Basically, you start with bunch of reference tiles (or
pieces) that can be put in a grid, as many times each as desired:
Collisions can also be added to the tiles, allowing for both 2D side scrolling and top down games.

**Making a tileset**

To begin, a tileset needs to be made. Here are some tiles for it. They are all in the same image because artists will often prefer this. Having them as separate images also works.

Create a new project and move the above png image into the directory.

We will be creating a *TileSet* resource. While this resource exports properties, it’s pretty difficult to get complex data into it and maintain it. Here is what it would look like to manually edit the resource:
Not the best way :)

TileSet

Property | Value
---|---
Name | floor
Texture | Texture image
Tex Offset | 0,0
Shape Offset | 0,0
Region | 0,0,64,64
Shape | tileset.xml

Name | edge
Texture | Texture image
Tex Offset | 0,0
Shape Offset | 0,0
Region | 64,0,64,64
Shape | tileset.xml

Name | wall
Texture | Texture image
Tex Offset | 0,0
Shape Offset | 0,0
Region | 64,64,64,64
There’s enough properties to get by, and with some effort editing this way can work, but the easiest way to edit and maintain a tileset is exporting it from a specially-crafted scene!

**TileSet scene**

Create a new scene with a regular node or node2d as root. For each tile, add a sprite as a child. Since tiles here are 50x50, enabling snap might be a good idea.

If more than one tile is present in the source image, make sure to use the region property of the sprite to adjust which part of the texture is being used.

Finally, make sure to name your sprite node correctly, this will ensure that, in subsequent edits to the tileset (for example, if added collision, changed the region, etc), the tile will still be **identified correctly and updated**. This name should be unique.

Sounds like a lot of requirements, so here’s a screenshot that shows where everything of relevance is:

![Image of TileSet scene in Godot Engine](image-url)
Continue adding all the tiles, adjusting the offsets if needed (if you have multiple tiles in a single source image). Again, remember that their names must be unique.

Collision

To add collision to a tile, create a StaticBody2D child for each sprite. This is a static collision node. Then, as a child of the StaticBody2D, create a CollisionShape2D or CollisionPolygon. The latter is recommended because it is easier to edit:

Finally, edit the polygon, this will give the tile a collision. **Remember to use snap!**. Using snap will make sure collision polygons are aligned properly, allowing a character to walk seamlessly from tile to tile. Also **do not scale or move** the collision and/or collision polygon nodes. Leave them at offset 0,0, with scale 1,1 and rotation 0 respect to the parent sprite.
Keep adding collisions to tiles until we are done. Note that BG is just a background, so it should not have a collision.

OK! We’re done! Remember to save this scene for future edit, call it “tileset_edit.scn” or something like that.
Exporting a TileSet

With the scene created and opened in the editor, next step will be to create a tileset. Use Scene > Convert To > Tile Set from the Scene Menu:

Then choose a filename, like “mytiles.res”. Make sure the “Merge With Existing” option is toggled on. This way, every time the tileset resource file is overwritten, existing tiles are merged and updated (they are referenced by their unique name, so again, **name your tiles properly**).

Using the TileSet in a TileMap

Create a new scene, use any node or node2d as root, then create a *TileMap* as a child.
Go to the tileset property of this node and assign the one created in previous steps:

Also set the cell size to ‘50’, since that is the size used by the tiles. Quadrant size is a tuning value, which means that the engine will draw and cull the tilemap in blocks of 16x16 tiles. This value is usually fine and does not need to be changed, but can be used to tune performance in specific cases (if you know what you are doing).

**Painting your world**

With all set, make sure the TileMap node is selected. A red grid will appear on screen, allowing to paint on it with the selected tile on the left palette.
To avoid moving and selecting the tilemap node accidentally (something common given it’s a huge node), it is recommended that you lock it, using the lock button:

You can also flip and rotate sprites in the TileMap editor (note: flipping the sprite in the TileSet will have no effect). Icons at the top right of the editor allow flipping and rotating of the currently selected sprite - you can also use A and S to flip horizontally and vertically. With a brick pattern like this tutorial uses, flipping the sprites would create unpleasant discontinuities unless you’re flipping an entire region of bricks, but for some kinds of tiles flipping can be a convenient and space-saving feature.

**Offset and scaling artifacts**

When using a single texture for all the tiles, scaling the tileset (or even moving to a non pixel-aligned location) will most likely result in filtering artifacts like this:
This can’t be avoided, as it is the way the hardware bilinear filter works. So, to avoid this situation, there are a few workarounds, try the ones that look better for you:

- Use a single image for each tile, this will remove all artifacts but can be more cumbersome to implement, so try the options below first.
- Disable filtering for either the tileset texture or the entire image loader (see the Managing image files asset pipeline tutorial).
- Enable pixel snap (set: “Scene > Project Settings > Display/use_2d_pixel_snap” to true).
- Viewport Scaling can often help with shrinking the map (see the Viewports tutorial).

### 4.2.4 Custom drawing in 2D

**Why?**

Godot has nodes to draw sprites, polygons, particles, and all sorts of stuff. For most cases this is enough, but not always. Before crying in fear, angst, and rage because a node to draw that-specific-something does not exist... it would be good to know that it is possible to easily make any 2D node (be it Control or Node2D based) draw custom commands. It is *really* easy to do it too.

**But...**

Custom drawing manually in a node is *really* useful. Here are some examples why:

- Drawing shapes or logic that is not handled by nodes (example: making a node that draws a circle, an image with trails, a special kind of animated polygon, etc).
- Visualizations that are not that compatible with nodes: (example: a tetris board). The tetris example uses a custom draw function to draw the blocks.
- Managing drawing logic of a large amount of simple objects (in the hundreds of thousands). Using a thousand nodes is probably not nearly as efficient as drawing, but a thousand of draw calls are cheap. Check the “Shower of Bullets” demo as example.
- Making a custom UI control. There are plenty of controls available, but it’s easy to run into the need to make a new, custom one.

**OK, how?**

Add a script to any CanvasItem derived node, like Control or Node2D. Override the _draw() function.
extends Node2D

func _draw():
    #your draw commands here
    pass

Draw commands are described in the CanvasItem class reference. There are plenty of them.

Updating

The _draw() function is only called once, and then the draw commands are cached and remembered, so further calls are unnecessary.

If re-drawing is required because a state or something else changed, simply call CanvasItem.update() in that same node and a new _draw() call will happen.

Here is a little more complex example. A texture variable that will be redrawn if modified:

extends Node2D

export var texture setget _set_texture

func _set_texture(value):
    #if the texture variable is modified externally,
    #this callback is called.
    texture=value #texture was changed
    update() #update the node

func _draw():
    draw_texture(texture,Vector2())

In some cases, it may be desired to draw every frame. For this, just call update() from the _process() callback, like this:

extends Node2D

func _draw():
    #your draw commands here
    pass

func _process(delta):
    update()

func _ready():
    set_process(true)

An example: drawing circular arcs

We will now use the custom drawing functionality of the Godot Engine to draw something that Godot doesn’t provide functions for. As an example, Godot provides a draw_circle() function that draws a whole circle. However, what about drawing a portion of a circle? You will have to code a function to perform this, and draw it yourself.
Arc function

An arc is defined by its support circle parameters, that is: the center position, and the radius. And the arc itself is then defined by the angle it starts from, and the angle it stops at. These are the 4 parameters we have to provide to our drawing. We’ll also provide the color value so we can draw the arc in different colors if we wish.

Basically, drawing a shape on screen requires it to be decomposed into a certain number of points linked one to the following one. As you can imagine, the more points your shape is made of, the smoother it will appear, but the heavier it will be in terms of processing cost. In general, if your shape is huge (or in 3D, close to the camera), it will require more points to be drawn without showing angular-looking. On the contrary, if you shape is small (or in 3D, far from the camera), you may reduce its number of points to save processing costs. This is called Level of Detail (LoD). In our example, we will simply use a fixed number of points, no matter the radius.

```godot
func draw_circle_arc( center, radius, angle_from, angle_to, color ):
    var nb_points = 32
    var points_arc = Vector2Array()
    for i in range(nb_points+1):
        var angle_point = angle_from + i*(angle_to-angle_from)/nb_points - 90
        var point = center + Vector2( cos(deg2rad(angle_point)), sin(deg2rad(angle_point)) ) * radius
        points_arc.push_back( point )
    for indexPoint in range(nb_points):
        draw_line(points_arc[indexPoint], points_arc[indexPoint+1], color)
```

Remember the number of points our shape has to be decomposed into? We fixed this number in the nb_points variable to a value of 32. Then, we initialize an empty Vector2Array, which is simply an array of Vector2.

Next step consists in computing the actual positions of these 32 points that compose arc. This is done in the first for-loop: we iterate over the number of points we want to compute the positions, plus one to include the last point. We first determine the angle of each point, between the starting and ending angles.

The reason why each angle is reduced of 90° is that we will compute 2D positions out of each angle using trigonometry (you know, cosine and sine stuff...). However, to be simple, cos() and sin() use radians, not degrees. The angle of 0° (0 radian) starts at 3 o’clock, although we want to start counting at 0 o’clock. So, we just reduce each angle of 90° in order to start counting from 0’clock.

The actual position of a point located on a circle at angle ‘angle’ (in radians) is given by Vector2(cos(angle), sin(angle)). Since cos() and sin() return values between -1 and 1, the position is located on a circle of radius 1. To have this position on our support circle, which has a radius of ‘radius’, we simply need to multiply the position by ‘radius’. Finally, we need to position our support circle at the ‘center’ position, which is performed by adding it to our Vector2 value. Finally, we insert the point in the Vector2Array which was previously defined.

Now, we need to actually draw our points. As you can imagine, we will not simply draw our 32 points: we need to draw everything that is between each of them. We could have computed every point ourselves using the previous method, and draw it one by one, but this it too complicated and inefficient (except if explicitly needed). So, we simply draw lines between each pair of points. Unless the radius of our support circle is very big, the length of each line between a pair of points will never be long enough to see them. If this happens, we simply would need to increase the number of points.

Draw the arc on screen

We now have a function that draws stuff on screen: it is time to call it in the _draw() function.
func _draw():
    var center = Vector2(200,200)
    var radius = 80
    var angle_from = 75
    var angle_to = 195
    var color = Color(1.0, 0.0, 0.0)
    draw_circle_arc( center, radius, angle_from, angle_to, color )

Result:

Arc polygon function

We can take this a step further and write a function that draws the plain portion of the disc defined by the arc, not only its shape. The method is exactly the same a previously, except that we draw a polygon instead of lines:

func draw_circle_arc_poly( center, radius, angle_from, angle_to, color ):
    var nb_points = 32
    var points_arc = Vector2Array()
    points_arc.push_back(center)
    var colors = ColorArray([color])
    for i in range(nb_points+1):
        var angle_point = angle_from + i*(angle_to-angle_from)/nb_points - 90
        points_arc.push_back(center + Vector2( cos( deg2rad(angle_point) ), sin(-deg2rad(angle_point) ) ) * radius)
    draw_polygon(points_arc, colors)
Dynamic custom drawing

Alright, we are now able to draw custom stuff on screen. However, it is very static: let’s make this shape turn around the center. The solution to do this is simply to change the angle_from and angle_to values over time. For our example, we will simply increment them by 50. This increment value has to remain constant, else the rotation speed will change accordingly.

First, we have to make both angle_from and angle_to variables global at the top of our script. Also note that you can store them in other nodes and access them using get_node().

```
extends Node2D
var rotation_ang = 50
var angle_from = 75
var angle_to = 195
```

We make these values change in the _process(delta) function. To activate this function, we need to call set_process(true) in the _ready() function.

We also increment our angle_from and angle_to values here. However, we must not forget to wrap() the resulting values between 0 and 360°! That is, if the angle is 361°, then it is actually 1°. If you don’t wrap these values, the script will work correctly but angles values will grow bigger and bigger over time, until they reach the maximum integer value Godot can manage (2^31 - 1). When this happens, Godot may crash or produce unexpected behavior. Since Godot doesn’t provide a wrap() function, we’ll create it here, as it is relatively simple.

Finally, we must not forget to call the update() function, which automatically calls _draw(). This way, you can control when you want to refresh the frame.

```
func _ready():
    set_process(true)
```
func wrap(value, min_val, max_val):
    var f1 = value - min_val
    var f2 = max_val - min_val
    return fmod(f1, f2) + min_val

func _process(delta):
    angle_from += rotation_ang
    angle_to += rotation_ang

    # we only wrap angles if both of them are bigger than 360
    if (angle_from > 360 && angle_to > 360):
        angle_from = wrap(angle_from, 0, 360)
        angle_to = wrap(angle_to, 0, 360)
    update()

Also, don’t forget to modify the _draw() function to make use of these variables:

func _draw():
    var center = Vector2(200,200)
    var radius = 80
    var color = Color(1.0, 0.0, 0.0)

    draw_circle_arc( center, radius, angle_from, angle_to, color )

Let’s run! It works, but the arc is rotating insanely fast! What’s wrong?

The reason is that your GPU is actually displaying the frames as fast as he can. We need to “normalize” the drawing by this speed. To achieve, we have to make use of the ‘delta’ parameter of the _process() function. ‘delta’ contains the time elapsed between the two last rendered frames. It is generally small (about 0.0003 seconds, but this depends on your hardware). So, using ‘delta’ to control your drawing ensures your program to run at the same speed on every hardware.

In our case, we simply need to multiply our ‘rotation_ang’ variable by ‘delta’ in the _process() function. This way, our 2 angles will be increased by a much smaller value, which directly depends on the rendering speed.

func _process(delta):
    angle_from += rotation_ang * delta
    angle_to += rotation_ang * delta

    # we only wrap angles if both of them are bigger than 360
    if (angle_from > 360 && angle_to > 360):
        angle_from = wrap(angle_from, 0, 360)
        angle_to = wrap(angle_to, 0, 360)
    update()

Let’s run again! This time, the rotation displays fine!

Tools

Drawing your own nodes might also be desired while running them in the editor, to use as preview or visualization of some feature or behavior.

Remember to just use the “tool” keyword at the top of the script (check the GDScript reference if you forgot what this does).
4.2.5 Particle Systems (2D)

Intro

A simple (but flexible enough for most uses) particle system is provided. Particle systems are used to simulate complex physical effects such as sparks, fire, magic particles, smoke, mist, magic, etc.

The idea is that a “particle” is emitted at a fixed interval and with a fixed lifetime. During his lifetime, every particle will have the same base behavior. What makes every particle different and provides a more organic look is the “randomness” associated to each parameter. In essence, creating a particle system means setting base physics parameters and then adding randomness to them.

Particles2D

Particle systems are added to the scene via the Particles2D node. They are enabled by default and start emitting white points downwards (as affected by the gravity). This provides a reasonable starting point to start adapting it to our needs.

Texture

A particle system uses a single texture (in the future this might be extended to animated textures via spritesheet). The texture is set via the relevant texture property:
Physics variables

Before taking a look at the global parameters for the particle system, let’s first see what happens when the physics variables are tweaked.

Direction

This is the base angle at which particles emit. Default is 0 (down):

Changing it will change the emissor direction, but gravity will still affect them:

This parameter is useful because, by rotating the node, gravity will also be rotated. Changing direction keeps them separate.

Spread

Spread is the angle at which particles will randomly be emitted. Increasing the spread will increase the angle. A spread of 180 will emit in all directions.

Linear velocity

Linear velocity is the speed at which particles will be emitted (in pixels/sec). Speed might later be modified by gravity or other accelerations (as described further below).

Spin velocity

Spin velocity is the speed at which particles turn around their center (in degrees/sec).

Orbit velocity

Orbit velocity is used to make particles turn around their center.

Gravity direction & strength

Gravity can be modified as in direction and strength. Gravity affects every particle currently alive.
Radial acceleration

If this acceleration is positive, particles are accelerated away from the center. If negative, they are absorbed towards it.

Tangential acceleration

This acceleration will use the tangent vector to the center. Combining with radial acceleration can do nice effects.

Damping

Damping applies friction to the particles, forcing them to stop. It is specially useful for sparks or explosions, which usually begin with a high linear velocity and then stop as they fade.

Initial angle

Determines the initial angle of the particle (in degrees). This parameter is mostly useful randomized.

Initial & final size

Determines the initial and final scales of the particle.

Color phases

Particles can use up to 4 color phases. Each color phase can include transparency. Phases must provide an offset value from 0 to 1, and always in ascending order. For example, a color will begin at offset 0 and end in offset 1, but 4 colors might use different offsets, such as 0, 0.2, 0.8 and 1.0 for the different phases:
Will result in:

**Global parameters**

These parameters affect the behavior of the entire system.

**Lifetime**

The time in seconds that every particle will stay alive. When lifetime ends, a new particle is created to replace it. Lifetime: 0.5

Lifetime: 4.0

**Timescale**

It happens often that the effect achieved is perfect, except too fast or too slow. Timescale helps adjust the overall speed. Timescale everything 2x:
Preprocess

Particle systems begin with 0 particles emitted, then start emitting. This can be an inconvenience when just loading a scene and systems like a torch, mist, etc begin emitting the moment you enter. Preprocess is used to let the system process a given amount of seconds before it is actually shown the first time.

Emit timeout

This variable will switch emission off after given amount of seconds being on. When zero, it’s disabled.

Offset

Allows to move the emission center away from the center

Half extents

Makes the center (by default 1 pixel) wider, to the size in pixels desired. Particles will emit randomly inside this area.

It is also possible to set an emission mask by using this value. Check the “Particles” menu on the 2D scene editor viewport and select your favorite texture. Opaque pixels will be used as potential emission location, while transparent ones will be ignored:

Local space

By default this option is on, and it means that the space that particles are emitted to is contained within the node. If the node is moved, all particles are moved with it:

If disabled, particles will emit to global space, meaning that if the node is moved, the emitter is moved too:

Explosiveness

If lifetime is 1 and there are 10 particles, it means every particle will be emitted every 0.1 seconds. The explosiveness parameter changes this, and forces particles to be emitted all together. Ranges are:

- 0: Emit all particles together.
- 1: Emit particles at equal interval.

Values in the middle are also allowed. This feature is useful for creating explosions or sudden bursts of particles:
Randomness

All physics parameters can be randomized. Random variables go from 0 to 1. The formula to randomize a parameter is:

\[
\text{initial\_value} = \text{param\_value} + \text{param\_value} \times \text{randomness}
\]

4.3 3D

4.3.1 Introduction to 3D

Creating a 3D game can be challenging. That extra Z coordinate makes many of the common techniques that helped to make 2D games simple no longer work. To aid in this transition, it is worth mentioning that Godot uses very similar APIs for 2D and 3D. Most nodes are the same and are present in both 2D and 3D versions. In fact, it is worth checking the 3D platformer tutorial, or the 3D kinematic character tutorials, which are almost identical to their 2D counterparts.

In 3D, math is a little more complex than in 2D, so also checking the Vector math in the wiki (which were specially created for game developers, not mathematicians or engineers) will help pave the way into efficiently developing 3D games.

Spatial node

Node2D is the base node for 2D. Control is the base node for everything GUI. Following this reasoning, the 3D engine uses the Spatial node for everything 3D.

Spatial nodes have a local transform, which is relative to the parent node (as long as the parent node is also or inherits of type Spatial). This transform can be accessed as a 4x3 Transform, or as 3 Vector3 members representing location, Euler rotation (x, y and z angles) and scale.
3D content

Unlike 2D, where loading image content and drawing is straightforward, 3D is a little more difficult. The content needs to be created with special 3D tool (usually referred to as DCCs) and exported to an exchange file format in order to be imported in Godot (3D formats are not as standardized as images).

DCC-created models

There are two pipelines to import 3D models in Godot. The first and most common one is through the Importing 3D scenes importer, which allows to import entire scenes (just as they look in the DCC), including animation, skeletal rigs, blend shapes, etc.

The second pipeline is through the Importing 3D meshes importer. This second method allows importing simple .OBJ files as mesh resources, which can be then put inside a MeshInstance node for display.

Generated geometry

It is possible to create custom geometry by using the Mesh resource directly, simply create your arrays and use the Mesh.add_surface() function. A helper class is also available, SurfaceTool, which provides a more straightforward API and helpers for indexing, generating normals, tangents, etc.

In any case, this method is meant for generating static geometry (models that will not be updated often), as creating vertex arrays and submitting them to the 3D API has a significant performance cost.

Immediate geometry

If, instead, there is a requirement to generate simple geometry that will be updated often, Godot provides a special node, ImmediateGeometry which provides an OpenGL 1.x style immediate-mode API to create points, lines, triangles, etc.

2D in 3D

While Godot packs a powerful 2D engine, many types of games use 2D in a 3D environment. By using a fixed camera (either orthogonal or perspective) that does not rotate, nodes such as Sprite3D and AnimatedSprite3D can be used to
Godot Engine Documentation, Release 2.1

create 2D games that take advantage of mixing with 3D backgrounds, more realistic parallax, lighting/shadow effects, etc.

The disadvantage is, of course, that added complexity and reduced performance in comparison to plain 2D, as well as the lack of reference of working in pixels.

**Environment**

Besides editing a scene, it is often common to edit the environment. Godot provides a `WorldEnvironment` node that allows changing the background color, mode (as in, put a skybox), and applying several types of built-in post-processing effects. Environments can also be overridden in the Camera.

**3D viewport**

Editing 3D scenes is done in the 3D tab. This tab can be selected manually, but it will be automatically enabled when a Spatial node is selected.

![3D viewport screenshot](image)

Default 3D scene navigation controls are similar to Blender (aiming to have some sort of consistency in the free software pipeline..), but options are included to customize mouse buttons and behavior to be similar to other tools in Editor Settings:
Coordinate system

Godot uses the metric system for everything. 3D Physics and other areas are tuned for this, so attempting to use a different scale is usually a bad idea (unless you know what you are doing).

When working with 3D assets, it’s always best to work in the correct scale (set your DCC to metric). Godot allows scaling post-import and, while this works in most cases, in rare situations it may introduce floating point precision issues (and thus, glitches or artifacts) in delicate areas such as rendering or physics. So, make sure your artists always work in the right scale!

The Y coordinate is used for “up”, though for most objects that need alignment (like lights, cameras, capsule collider, vehicle, etc.), the Z axis is used as a “pointing towards” direction. This convention roughly means that:

- X is sides
- Y is up/down
- Z is front/back

Space and manipulation gizmos

Moving objects in the 3D view is done through the manipulator gizmos. Each axis is represented by a color: Red, Green, Blue represent X,Y,Z respectively. This convention applies to the grid and other gizmos too (and also to the shader language, ordering of components for Vector3,Color,etc.).
Some useful keybindings:

- To snap motion or rotation, press the ‘s’ key while moving, scaling or rotating.
- To center the view on the selected object, press the ‘f’ key.

View menu

The view options are controlled by the “[ view ]” menu. Pay attention to this little menu inside the window because it is often overlooked!
Default lighting

The 3D view has by some default options on lighting:

- There is a directional light that makes objects visible while editing turned on by default. It is no longer visible when running the game.
- There is subtle default environment light to avoid places not reached by the light to remain visible. It is also no longer visible when running the game (and when the default light is turned off).

These can be turned off by toggling the “Default Light” option:

Customizing this (and other default view options) is also possible via the settings menu:
Which opens this window, allowing to customize ambient light color and default light direction:
Cameras

No matter how many objects are placed in 3D space, nothing will be displayed unless a *Camera* is also added to the scene. Cameras can either work in orthogonal or perspective projections:
Cameras are associated and only display to a parent or grand-parent viewport. Since the root of the scene tree is a viewport, cameras will display on it by default, but if sub-viewport (either as render target or picture-in-picture) are desired, they need their own children cameras to display.

When dealing with multiple cameras, the following rules are followed for each viewport:

- If no cameras are present in the scene tree, the first one that enters it will become the active camera. Further cameras entering the scene will be ignored (unless they are set as current).
• If a camera has the “current” property set, it will be used regardless of any other camera in the scene. If the property is set, it will become active, replacing the previous camera.

• If an active camera leaves the scene tree, the first camera in tree-order will take its place.

Lights

There is no limitation on the number of lights nor of types of lights in Godot. As many as desired can be added (as long as performance allows). Shadow maps are, however, limited. The more they are used, the less the quality overall. It is possible to use doc_light_baking, to avoid using large amount of real-time lights and improve performance.

4.3.2 3D performance and limitations

Introduction

Godot follows a balanced performance philosophy. In performance world, there are always trade-offs, which consist in trading speed for usability and flexibility. Some practical examples of this are:

• Rendering objects efficiently in high amounts is easy, but when a large scene must be rendered it can become inefficient. To solve this, visibility computation must be added to the rendering, which makes rendering less efficient, but at the same time less objects are rendered, so efficiency overall improves.

• Configuring the properties of every material for every object that needs to be rendered is also slow. To solve this, objects are sorted by material to reduce the costs, but at the same time sorting has a cost.

• In 3D physics a similar situation happens. The best algorithms to handle large amounts of physics objects (such as SAP) are very slow at insertion/removal of objects and ray-casting. Algorithms that allow faster insertion and removal, as well as ray-casting will not be able to handle as many active objects.

And there are many more examples of this! Game engines strive to be general purpose in nature, so balanced algorithms are always favored over algorithms that might be the fast in some situations and slow in others.. or algorithms that are fast but make usability more difficult.

Godot is not an exception and, while it is designed to have backends swappable for different algorithms, the default ones (or more like, the only ones that are there for now) prioritize balance and flexibility over performance.

With this clear, the aim of this tutorial is to explain how to get the maximum performance out of Godot.

Rendering

3D rendering is one of the most difficult areas to get performance from, so this section will have a list of tips.

Reuse shaders and materials

Godot renderer is a little different to what is out there. It’s designed to minimize GPU state changes as much as possible. FixedMaterial does a good job at reusing materials that need similar shaders but, if custom shaders are used, make sure to reuse them as much as possible. Godot’s priorities will be like this:

• Reusing Materials: The less amount of different materials in the scene, the faster the rendering will be. If a scene has a huge amount of objects (in the hundreds or thousands) try reusing the materials or in the worst case use atlases.

• Reusing Shaders: If materials can’t be reused, at least try to re-use shaders (or FixedMaterials with different parameters but same configuration).
If a scene has, for example, 20,000 objects with 20,000 different materials each, rendering will be really slow. If the same scene has 20,000 objects, but only uses 100 materials, rendering will be blazing fast.

**Pixels cost vs vertex cost**

It is a common thought that the lower the polygons in a model, the faster it will be rendered. This is really relative and depends on many factors.

On a modern PC and consoles, vertex cost is low. Very low. GPUs originally only rendered triangles, so all the vertices:

1. Had to be transformed by the CPU (including clipping).
2. Had to be sent to the GPU memory from the main RAM.

Nowadays, all this is handled inside the GPU, so the performance is extremely high. 3D artists usually have the wrong feeling about polycount performance because 3D DCCs (such as Blender, Max, etc.) need to keep geometry in CPU memory in order for it to be edited, reducing actual performance. Truth is, a model rendered by a 3D engine is much more optimal than how 3D DCCs display them.

On mobile devices, the story is different. PC and Console GPUs are brute-force monsters that can pull as much electricity as they need from the power grid. Mobile GPUs are limited to a tiny battery, so they need to be a lot more power efficient.

To be more efficient, mobile GPUs attempt to avoid overdraw. This means, the same pixel on the screen being rendered (as in, with lighting calculation, etc.) more than once. Imagine a town with several buildings, GPUs don’t really know what is visible and what is hidden until they draw it. A house might be drawn and then another house in front of it (rendering happened twice for the same pixel!). PC GPUs normally don’t care much about this and just throw more pixel processors to the hardware to increase performance (but this also increases power consumption).

On mobile, pulling more power is not an option, so a technique called “Tile Based Rendering” is used (almost every mobile hardware uses a variant of it), which divide the screen into a grid. Each cell keeps the list of triangles drawn to it and sorts them by depth to minimize overdraw. This technique improves performance and reduces power consumption, but takes a toll on vertex performance. As a result, less vertices and triangles can be processed for drawing.

Generally, this is not so bad, but there is a corner case on mobile that must be avoided, which is to have small objects with a lot of geometry within a small portion of the screen. This forces mobile GPUs to put a lot of strain on a single screen cell, considerably decreasing performance (as all the other cells must wait for it to complete in order to display the frame).

To make it short, do not worry about vertex count so much on mobile, but avoid concentration of vertices in small parts of the screen. If, for example, a character, NPC, vehicle, etc. is far away (so it looks tiny), use a smaller level of detail (LOD) model instead.

An extra situation where vertex cost must be considered is objects that have extra processing per vertex, such as:

- Skinning (skeletal animation)
- Morphs (shape keys)
- Vertex Lit Objects (common on mobile)

**Texture compression**

Godot offers to compress textures of 3D models when imported (VRAM compression). Video RAM compression is not as efficient in size as PNG or JPG when stored, but increase performance enormously when drawing.

This is because the main goal of texture compression is bandwidth reduction between memory and the GPU.
In 3D, the shapes of objects depend more on the geometry than the texture, so compression is generally not noticeable. In 2D, compression depends more on shapes inside the textures, so the artifacting resulting from the compression is more noticeable.

As a warning, most Android devices do not support texture compression of textures with transparency (only opaque), so keep this in mind.

**Transparent objects**

As mentioned before, Godot sorts objects by material and shader to improve performance. This, however, can not be done on transparent objects. Transparent objects are rendered from back to front to make blending with what is behind work. As a result, please try to keep transparent objects to a minimum! If an object has a small section with transparency, try to make that section a separate material.

**Level of detail (LOD)**

As also mentioned before, using objects with less vertices can improve performance in some cases. Godot has a very simple system to use level of detail, *GeometryInstance* based objects have a visibility range that can be defined. Having several GeometryInstance objects in different ranges works as LOD.

**Use instancing (MultiMesh)**

If several identical objects have to be drawn in the same place or nearby, try using *MultiMesh* instead. MultiMesh allows drawing of dozens of thousands of objects at very little performance cost, making it ideal for flocks, grass, particles, etc.

**Bake lighting**

Small lights are usually not a performance issue. Shadows a little more. In general, if several lights need to affect a scene, it’s ideal to bake it (doc_light_baking). Baking can also improve the scene quality by adding indirect light bounces.

If working on mobile, baking to texture is recommended, since this method is even faster.

**4.3.3 Importing 3D meshes**

**Introduction**

Godot supports a flexible and powerful 3D scene importer that allows for full scene importing. For a lot of artists and developers this is more than enough. However, many do not like this workflow as much and prefer to import individual 3D Meshes and build the scenes inside the Godot 3D editor themselves. (Note that for more advanced features such as skeletal animation, there is no option to the 3D Scene Importer).

The 3D mesh import workflow is simple and works using the OBJ file format. The imported meshes result in a .msh binary file which the user can put into a *MeshInstance*, which in turn can be placed somewhere in the edited scene.
Importing

Importing is done through the Import 3D Mesh menu:

Which opens the Mesh import window:
This dialog allows the import of one more more OBJ files into a target path. OBJ files are converted to .msh files. Files are imported without any material on them, material has to be added by the user (see the Fixed materials tutorial). If the external OBJ file is changed it will be re-imported, while keeping the newly assigned material.

**Options**

A few options are present. Normals are needed for regular shading, while Tangents are needed if you plan to use normal-mapping on the material. In general, OBJ files describe how to be shaded very well, but an option to force smooth shading is available.

Finally, there is an option to weld vertices. Given OBJ files are text-based, it is common to find some of these with vertices that do not match, which results in strange shading. The weld vertices option merges vertices that are too close to keep proper smooth shading.
Usage

Mesh resources (what this importer imports to) are used inside MeshInstance nodes. Simply set them to the Mesh property of them.

And that is it.

4.3.4 Importing 3D scenes

Introduction

Most game engines just import 3D objects, which may contain skeletons or animations, and then all further work is done in the engine UI, like object placement, full scene animations, etc. In Godot, given the node system is very similar to how 3D DCC tools (such as Maya, 3DS Max or Blender) work, full 3D scenes can be imported in all their glory. Additionally, by using a simple language tag system, it is possible to specify that objects are imported as several things, such as collidable, rooms and portals, vehicles and wheels, LOD distances, billboards, etc.

This allows for some interesting features:

- Importing simple scenes, rigged objects, animations, etc.
- Importing full scenes. Entire scenarios can be created and updated in the 3D DCC and imported to Godot each time they change, then only little editing is needed from the engine side.
- Full cutscenes can be imported, including multiple character animation, lighting, camera motion, etc.
- Scenes can be further edited and scripted in the engine, where shaders and environment effects can be added, enemies can be instanced, etc. The importer will update geometry changes if the source scene changes but keep the local changes too (in real-time while using the Godot editor!)
- Textures can be all batch-imported and updated when the source scene changes.

This is achieved by using a very simple language tag that will be explained in detail later.
Exporting DAE files

Why not FBX?

Most game engines use the FBX format for importing 3D scenes, which is definitely one of the most standardized in the industry. However, this format requires the use of a closed library from Autodesk which is distributed with a more restrictive licensing terms than Godot. The plan is, sometime in the future, to implement an external conversion binary, but meanwhile FBX is not really supported.

Exporting DAE files from Maya and 3DS Max

Autodesk added built-in collada support to Maya and 3DS Max, but it’s really broken and should not be used. The best way to export this format is by using the OpenCollada plugins. They work really well, although they are not always up-to-date with the latest version of the software.

Exporting DAE files from Blender

Blender also has built-in collada support, but it’s really broken and should not be used either.

Godot provides a Python Plugin that will do a much better job at exporting the scenes.

The import process

Import process begins with the 3D scene import menu:

That opens what is probably the biggest of all the import dialogs:
Many options exist in there, so each section will be explained as follows:

### Source & target paths

To import, two options are needed. The first is a source .dae file (“Digital Asset Exchange”, the extension of the Collada standard). More import formats will eventually added, but Collada is the most complete open format as of this writing.

A target folder needs to be provided, so the importer can import the scene there. The imported scene will have the same filename as the source one, except for the .scn extension, so make sure you pick good names when you export!

The textures will be copied and converted. Textures in 3D applications are usually just PNG or JPG files. Godot will convert them to video memory texture compression format (s3tc, pvrtc, ericsson, etc.) by default to improve performance and save resources.

Since the original textures, 3D file and textures are usually not needed, it’s recommended to keep them outside the project. For some hints on how to do this the best way, you can check the Project organization tutorial.

Two options for textures are provided. They can be copied to the same place as the scene, or they can be copied to a common path (configurable in the project settings). If you choose this, make sure no two textures are named the same.

### 3D rigging tips

Before going into the options, here are some tips for making sure your rigs import properly
• Only up to 4 weights are imported per vertex, if a vertex depends of more than 4 bones, only the 4 most important bones (the one with the most weight) will be imported. For most models this usually works fine, but just keep it in mind.

• Do not use non-uniform scale in bone animation, as this will likely not import properly. Try to accomplish the same effect with more bones.

• When exporting from Blender, make sure that objects modified by a skeleton are children of it. Many objects can be modified by a single skeleton, but they all should be direct children.

• The same way, when using Blender, make sure that the relative transform of children nodes to the skeleton is zero (no rotation, no translation, no scale. All zero and scale at 1.0). The position of both objects (the little orange dot) should be at the same place.

3D import options

This section contains many options to change the way import workflow works. Some (like HDR) will be better explained in other sections, but in general a pattern can be visible in the options and that is, many of the options end with “-something”. For example:

• Remove Nodes (-noimp)
• Set Alpha in Materials (-alpha)
• Create Collisions (-col).

This means that the object names in the 3D DCC need to have those options appended at the end for the importer to tell what they are. When imported, Godot will convert them to what they are meant to be.

Note: Maya users must use “_” (underscore) instead of “-” (minus).

Here is an example of how a scene in the 3D DCC looks (using Blender), and how it is imported to Godot:

Notice that:

• The camera was imported normally.
• A Room was created (-room).
• A Portal was created (-portal).
• The Mesh got static collision added (-col).
• The Light was not imported (-noimp).
Options in detail

Following is a list of most import options and what they do in more detail.

Remove nodes (-noimp)

Node names that have this at the end will be removed at import time, no matter their type. Erasing them afterwards is most of the times pointless because the will be restored if the source scene changes.

Import animations

Some scene formats (.dae) support one or more animations. If this is checked, an AnimationPlayer node will be created, containing the animations.

Compress geometry

This option (disabled [STRIKEOUT: or more like, always enabled] at the moment at the time of writing this) will compress geometry so it takes less space and renders faster (at the cost of less precision).

Force generation of tangent arrays

The importer detects when you have used a normalmap texture, or when the source file contains tangent/binormal information. These arrays are needed for normalmapping to work, and most exporters know what they do when they export this. However, it might be possible to run into source scenes that do not have this information which, as a result, make normal-mapping not work. If you notice that normal-maps do not work when importing the scene, turn this on!

SRGB -> linear of diffuse textures

When rendering using HDR (High Dynamic Range) it might be desirable to use linear-space textures to achieve a more real-life lighting. Otherwise, colors may saturate and contrast too much when exposure changes. This option must be used together with the SRGB option in WorldEnvironment. The texture import options also have the option to do this conversion, but if this one is turned on, conversion will always be done to diffuse textures (usually what is desired). For more information, read the High dynamic range tutorial.

Set alpha in materials (-alpha)

When working with most 3D DCCs, it's pretty obvious when a texture is transparent and has opacity and this rarely affects the workflow or final rendering. However, when dealing with real-time rendering, materials with alpha blending are usually less optimal to draw, so they must be explicitly marked as such.

Originally Godot detected this based on whether if the source texture had an alpha channel, but most image manipulation applications like Photoshop or Gimp will export this channel anyway even if not used. Code was added later to check manually if there really was any transparency in the texture, but artists will anyway and very often lay uvmaps into opaque parts of a texture and leave unused areas (where no UV exists) transparent, making this detection worthless.

Finally, it was decided that it's best to import everything as opaque and leave artists to fix materials that need transparency when it's obvious that they are not looking right (see the Materials tutorial).
As a helper, since every 3D DCC allows naming the materials and keeping their name upon export, the (-alpha) modifier in their name will hint the 3D scene importer in Godot that this material will use the alpha channel for transparency.

**Set vert. color in materials (-vcol)**

Most 3D DCCs support vertex color painting. This is generally applied as multiplication or screen blending. However, it is also often the case that your exporter will export this information as all 1s, or export it as something else and you will not realize it. Since most of the cases this option is not desired, just add this to any material to confirm that vertex colors are desired.

**Create collisions (-col, -colonly)**

Option “-col” will work only for Mesh nodes. If it is detected, a child static collision node will be added, using the same geometry as the mesh.

However, it is often the case that the visual geometry is too complex or too un-smooth for collisions, which end up not working well. To solve this, the “-colonly” modifier exists, which will remove the mesh upon import and create a StaticBody collision instead. This helps the visual mesh and actual collision to be separated.

Option “-colonly” can be also used with Blender’s empty objects. On import it will create a StaticBody with collision node as a child. Collision node will have one of predefined shapes, depending on the Blender’s empty draw type:

- Single arrow will create RayShape
- Cube will create BoxShape
- Image will create PlaneShape
- Sphere (and other non-listed) will create SphereShape

For better visibility in Blender’s editor user can set “X-Ray” option on collision empties and set some distinct color for them in User Preferences / Themes / 3D View / Empty.
Create rooms (-room)

This is used to create a room. As a general rule, any node that is a child of this node will be considered inside the room (including portals).

There are two ways in which this modifier can be used. The first is using a Dummy/Empty node in the 3D application with the “-room” tag. For this to work, the “interior” of the room must be closed (geometry of the children should contain walls, roof, floor, etc. and the only holes to the outside should be covered with portals). The importer will then create a simplified version of the geometry for the room.

The second way is to use the “-room” modifier on a mesh node. This will use the mesh as the base for the BSP tree that contains the room bounds. Make sure that the mesh shape is closed, all normals point outside and that the geometry is not self-intersecting, otherwise the bounds may be computed wrong (BSP Trees are too picky and difficult to work with, which is why they are barely used anymore..).

Anyway, the room will need portals, which are described next.

Create portals (-portal)

Portals are the view to look outside a room. They are always some flat shape on the surface of a room. If the portal is left alone, it is used to activate occlusion when looking inside<->outside the room.

Basically, the conditions to make and import a portal from the 3D DCC are:

- It should be a child of a room.
- It should lay on the surface of the room (this doesn’t need to be super exact, just make it as close as you can by eye and Godot will adjust it)
- It must be a flat, convex shape, any flat and convex shape is okay, no matter the axis or size.
- Normals for the flat shape faces must all point towards the OUTSIDE of the room.

Here is how it usually looks:
To connect to rooms, simply make two identical portals for both rooms and place them overlapped. This does not need to be perfectly exact, again, as Godot will fix it.

[...]

The rest of the tags in this section should be rather obvious, or will be documented/changed in the future.

**Double-sidedness**

Collada and other formats support specifying the double-sidedness of the geometry (in other words, when not double-sided, back-faces are not drawn). Godot supports this option per Material, not per Geometry.

When exporting from 3D DCCs that work with per-object double-sidedness (such as Blender of Maya), make sure that the double sided objects do not share a material with the single sided ones or the importer will not be able to discern.

**Animation options**

Some things to keep in mind when importing animations. 3D DCCs allow animating with curves for every x,y,z component, doing IK constraints and other stuff. When imported for real-time, animations are sampled (at small intervals) so all this information is lost. Sampled animations are fast to process, but can use considerable amounts of memory.

Because of this, the “Optimize” option exists but, in some cases, this option might break an animation, so make it sure to disable it if you notice any issues.
Some animations are meant to be cycled (like walk animations) if this is the case, animation names that end in “-cycle” or “-loop” are automatically set to loop.

Import script

Creating a script to parse the imported scene is actually really simple. This is great for post processing, changing materials, doing funny stuff with the geometry, etc.

Create a script that basically looks like this:

```godot
tool # needed so it runs in editor
extends EditorScenePostImport

func post_import(scene):
    # do your stuff here
    return scene # remember to return the imported scene
```

The post-import function takes the imported scene as parameter (the parameter is actually the root node of the scene).

Update logic

Other types of resources (like samples, meshes, fonts, images, etc.) are re-imported entirely when changed and user changes are not kept.

Because of 3D Scenes can be really complex, they use a different update strategy. The user might have done local changes to take advantage of the engine features and it would be really frustrating if everything is lost on re-import because the source asset changed.

This led to the implementation of a special update strategy. The idea behind is that the user will not lose anything he or she did, and only added data or data that can’t be edited inside Godot will be updated.

It works like this:

Strategy

Upon changes on the source asset (ie: .dae), and on re-import, the editor will remember the way the scene originally was, and will track your local changes like renaming nodes, moving them or reparenting them. Finally, the following will be updated:

- Mesh Data will be replaced by the data from the updated scene.
- Materials will be kept if they were not modified by the user.
- Portal and Room shapes will be replaced by the ones from the updated scene.
- If the user moved a node inside Godot, the transform will be kept. If the user moved a node in the source asset, the transform will be replaced. Finally, if the node was moved in both places, the transform will be combined.

In general, if the user deletes anything from the imported scene (node, mesh, material, etc.), updating the source asset will restore what was deleted. This is a good way to revert local changes to anything. If you really don’t want a node anymore in the scene, either delete it from both places or add the “-noimp” tag to it in the source asset.
Fresh re-import

It can also happen that the source asset changed beyond recognition and a full fresh re-import is desired. If so, simply re-open the 3D scene import dialog from the Import -> Re-Import menu and perform re-import.

4.3.5 Materials

Introduction

Materials can be applied to most visible 3D objects. Basically they describe how light interacts with that object. There are many types of materials, but the main ones are the FixedMaterial and the ShaderMaterial. Tutorials for each of them exist Fixed materials and Shader materials.

This tutorial is about the basic properties shared between them.

Flags

Materials, no matter which type they are, have an associated set of flags. Their use will be explained in the following.

Visible

Toggles whether the material is visible. If unchecked, the object will not be shown.

Double sided & inverted faces

Godot by default only shows geometry faces (triangles) when their front-side faces the camera. If looking at the front-side of a face, its vertices have to be oriented clockwise by definition. For closed objects, the back-side of faces is never visible because they are hidden by other faces. SO not drawing invisible triangles (whose vertices are oriented counter-clockwise on the view plane) saves a lot of GPU power.

However, for flat or open objects, the back-side of faces might be visible and needs to be drawn as well. The “double sided” flag makes sure that no matter the facing, the triangle will always be drawn. It is also possible to invert this
check and draw only counter-clockwise looking faces, though it’s not very useful except for a few cases (like drawing outlines).

**Unshaded**

Objects are always black unless light affects them, and their shading changes according to the type and direction of lights. When this flag is turned on, the diffuse color is displayed right the same as it appears in the texture or parameter:

![shaded and unshaded objects]

**On top**

When this flag is turned on, the object will be drawn after everything else has been drawn and without a depth test. This is generally useful for objects which shall never be hidden by other objects such as HUD effects or gizmos.

**Lightmap on UV2**

When using lightmapping (see the doc_light_baking tutorial), this option determines that the lightmap should be accessed on the UV2 array instead of regular UV.

**Parameters**

Some parameters also exist for controlling drawing and blending:
Blend mode

Objects are usually blended in Mix mode. Other blend modes (Add and Sub) exist for special cases (usually particle effects, light rays, etc.) but materials can be set to them:

![Blend modes](image.png)

Line width

When drawing lines, the size of them can be adjusted here per material.

Depth draw mode

This is a tricky but very useful setting. By default, opaque objects are drawn using the depth buffer and translucent objects are not (but are sorted by depth). This behavior can be changed here. The options are:

- **Always**: Draw objects with depth always, even those with alpha. This often results in glitches like the one in the first image (which is why it’s not the default).

- **Opaque Only**: Draw objects with depth only when they are opaque, and do not set depth for alpha. This is the default because it’s fast, but it’s not the most correct setting. Objects with transparency that self-intersect will always look wrong, especially those that mix opaque and transparent areas, like grass, tree leaves, etc. Objects with transparency also can’t cast shadows, this is evident in the second image.

- **Alpha Pre-Pass**: The same as above, but a depth pass is performed for the opaque areas of objects with transparency. This makes objects with transparency look much more correct. In the third image it is evident how the leaves cast shadows between them and into the floor. This setting is turned off by default because, while on PC this is not very costly, mobile devices suffer a lot when this setting is turned on, so use it with care.

- **Never**: Never use the depth buffer for this material. This is mostly useful in combination with the “On Top” flag explained above.
depth draw always

depth draw opaque only

depth draw pre-pass alpha
4.3.6 Fixed materials

Introduction

Fixed materials (originally Fixed Pipeline Materials) are the most common type of materials, using the most common material options found in 3D DCCs (such as Maya, 3DS Max or Blender). The big advantage of using them is that 3D artists are very familiar with this layout. They also allow to try out different things quickly without the need of writing shaders. Fixed Materials inherit from Material, which also has several options. If you haven’t read it before, reading the Materials tutorial is recommended.

Options

Here is the list of all the options available for fixed materials:
### Fixed Material

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Fixed Flags</strong></td>
<td></td>
</tr>
<tr>
<td>Use Alpha</td>
<td>On</td>
</tr>
<tr>
<td>Use Color Array</td>
<td>On</td>
</tr>
<tr>
<td>Use Point Size</td>
<td>On</td>
</tr>
<tr>
<td>Discard Alpha</td>
<td>On</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Params</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Diffuse</td>
<td></td>
</tr>
<tr>
<td>Specular</td>
<td></td>
</tr>
<tr>
<td>Emission</td>
<td></td>
</tr>
<tr>
<td>Specular Exp</td>
<td>40</td>
</tr>
<tr>
<td>Detail Blend</td>
<td>Mix</td>
</tr>
<tr>
<td>Detail Mix</td>
<td>1</td>
</tr>
<tr>
<td>Normal Depth</td>
<td>1</td>
</tr>
<tr>
<td>Shade Param</td>
<td>0.5</td>
</tr>
<tr>
<td>Glow</td>
<td>0</td>
</tr>
<tr>
<td>Point Size</td>
<td>1</td>
</tr>
<tr>
<td>Blend Mode</td>
<td>Mix</td>
</tr>
<tr>
<td>Line Width</td>
<td>0.1</td>
</tr>
<tr>
<td>Uv Xform</td>
<td>1, 0, 0, 0, 1, 0,</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Textures</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Diffuse</td>
<td>&lt;null&gt;</td>
</tr>
<tr>
<td>Diffuse Tc</td>
<td>UV</td>
</tr>
<tr>
<td>Detail</td>
<td>&lt;null&gt;</td>
</tr>
<tr>
<td>Detail Tc</td>
<td>UV</td>
</tr>
<tr>
<td>Specular</td>
<td>&lt;null&gt;</td>
</tr>
<tr>
<td>Specular Tc</td>
<td>UV</td>
</tr>
<tr>
<td>Emission</td>
<td>&lt;null&gt;</td>
</tr>
<tr>
<td>Emission Tc</td>
<td>UV</td>
</tr>
<tr>
<td>Specular Exp</td>
<td>&lt;null&gt;</td>
</tr>
<tr>
<td>Specular Exp T</td>
<td>UV</td>
</tr>
<tr>
<td>Glow</td>
<td>&lt;null&gt;</td>
</tr>
<tr>
<td>Glow Tc</td>
<td>UV</td>
</tr>
<tr>
<td>Normal</td>
<td>&lt;null&gt;</td>
</tr>
<tr>
<td>Normal Tc</td>
<td>UV</td>
</tr>
<tr>
<td>Shade Param</td>
<td>&lt;null&gt;</td>
</tr>
<tr>
<td>Shade Param T</td>
<td>UV</td>
</tr>
</tbody>
</table>
From this point, every option will be explained in detail:

**Fixed flags**

These are a set of flags that control general aspects of the material.

**Use alpha**

This flag needs to be active for transparent materials to blend with what is behind, otherwise display will always be opaque. Do not enable this flag unless the material really needs it, because it can severely affect performance and quality. Materials with transparency also won’t cast shadows (unless they contain opaque areas and the “opaque pre-pass” hint is turned on, see the Materials tutorial for more information).

![alpha disabled](image)

![alpha enabled](image)

**Use vertex colors**

Vertex color painting is a very common technique to add detail to a geometry. 3D DCCs all support this, and many even support baking occlusion to it. Godot allows this information to be used in the fixed material by modulating the diffuse color when enabled.

![vertex colors disabled](image)

![vertex colors enabled](image)
**Point size**

Point size is used to set the point size (in pixels) for when rendering points. This feature is mostly used in tools and HUDs.

**Discard alpha**

When alpha is enabled (see above) the transparent pixels are blended with what is behind them. In some combinations (of using alpha to render depth) it may be possible that transparent pixels cover other objects.

If this is the case, enable this option for the material. This option is often used in combination with “opaque pre-pass” hint (see the *Materials* tutorial for more information).

**Parameters**

**Diffuse, specular, emission and specular exponent**

These are the base colors for the material.

- Diffuse color is responsible for the light that reaches the material, and is diffusely back-scattered then. This color varies by the angle between the face and the light and the distance to the light source (in the case of spot and omni lights). It is the color that best represents the material. It can also have alpha (transparency).

- Specular color is the color of the reflected light and responsible for shines. It is affected by the specular exponent.

- Emission is the color of the light generated within the material (although it will not lit anything else around unless baking). This color is constant.

- Specular Exponent (or “Shininess”/”Intensity” in some 3D DCCs) is the way light is reflected. If the value is high, light is reflected completely, otherwise it is diffused more and more.

Below is an example of how they interact:
Shader & shader param

Regular shader materials allow custom lighting code. Fixed materials come with four predefined shader types:

- **Lambert**: The standard diffuse light, where the amount of light is proportional to the angle from the light emitter.
- **Wrap**: A variation on Lambert, where the “coverage” of the light can be changed. This is useful for many types of materials such as wood, clay, hair, etc.
- **Velvet**: This is similar to Lambert, but adds light scattering in the edges. It’s useful for leathers and some types of metals.
- **Toon**: Standard toon shading with a coverage parameter. The specular component also becomes toon-ized.
Detail & detail mix

Detail is a second diffuse texture which can be mixed with the first one (more on textures later!). Detail blend and mix control how these are added together, here’s an example of what detail textures are for:
Normal depth

Normal depth controls the strength and the direction of normal-mapping. If it is set to 1 (the default), the un-scaled normal map is applied. Values larger than 1 make normal-mapping more pronounced (dents and bumps become larger), while values smaller than 1 reduce the effect. A normal depth of 0 disables normal-mapping. Negative values invert the normal map so dents become bumps and vice versa. Here is an example showing the influence of the normal depth on the outcome:
Glow

This value controls how much of the color is sent to the glow buffer. It can be greater than 1 for a stronger effect. For glow to work, a WorldEnvironment must exist with Glow activated.
Blend mode

Objects are usually blended in Mix mode. Other blend modes (Add and Sub) exist for special cases (usually particle effects, light rays, etc.) but materials can be set to them:

Point size, line width

When drawing points or lines, the size of them can be adjusted here per material.

Textures

Almost all of the parameters above can have a texture assigned to them. There are four options to where they can get their UV coordinates:

- **UV Coordinates (UV Array)**: This is the regular UV coordinate array that was imported with the model.
- **UV x UV XForm**: UV Coordinates multiplied by the UV Xform matrix.
- **UV2 Coordinates**: Some imported models might have come with a second set of UV coordinates. These are common for detail textures or for baked light textures.
- **Sphere**: Spherical coordinates (difference of the normal at the pixel by the camera normal).

The value of every pixel of the texture is multiplied by the original parameter. This means that if a texture is loaded for diffuse, it will be multiplied by the color of the diffuse color parameter. Same applies to all the others except for specular exponent, which is replaced.
4.3.7 High dynamic range

Introduction

Normally, an artist does all the 3D modelling, then all the texturing, looks at his or her awesome looking model in the 3D DCC and says “looks fantastic, ready for integration!” then goes into the game, lighting is setup and the game runs.

So where does all this HDR stuff thing come from? The idea is that instead of dealing with colors that go from black to white (0 to 1), we use colors whiter than white (for example, 0 to 8 times white).

To be more practical, imagine that in a regular scene, the intensity of a light (generally 1.0) is set to 5.0. The whole scene will turn very bright (towards white) and look horrible.

After this the luminance of the scene is computed by averaging the luminance of every pixel of it, and this value is used to bring the scene back to normal ranges. This last operation is called tone-mapping. Finally, we are at a similar place from where we started:
Except the scene is more contrasted, because there is a higher light range in play. What is this all useful for? The idea is that the scene luminance will change while you move through the world, allowing situations like this to happen:
Additionally, it is possible to set a threshold value to send to the glow buffer depending on the pixel luminance. This allows for more realistic light bleeding effects in the scene.

Linear color space

The problem with this technique is that computer monitors apply a gamma curve to adapt better to the way the human eye sees. Artists create their art on the screen too, so their art has an implicit gamma curve applied to it.

The color space where images created in computer monitors exist is called “sRGB”. All visual content that people have on their computers or download from the internet (such as pictures, movies, etc.) is in this colorspace.

The mathematics of HDR require that we multiply the scene by different values to adjust the luminance and exposure to different light ranges, and this curve gets in the way as we need colors in linear space for this.

Linear color space & asset pipeline

Working in HDR is not just pressing a switch. First, imported image assets must be converted to linear space on import. There are two ways to do this:

SRGB -> linear conversion on image import

This is the most compatible way of using linear-space assets and it will work everywhere including all mobile devices. The main issue with this is loss of quality, as sRGB exists to avoid this same problem. Using 8 bits per channel to represent linear colors is inefficient from the point of view of the human eye. These textures might be later compressed too, which makes the problem worse.

In any case though, this is the easy solution that works everywhere.

Hardware sRGB -> linear conversion.

This is the most correct way to use assets in linear-space, as the texture sampler on the GPU will do the conversion after reading the texel using floating point. This works fine on PC and consoles, but most mobile devices do no support it, or do not support it on compressed texture format (iOS for example).
Linear -> sRGB at the end.

After all the rendering is done, the linear-space rendered image must be converted back to sRGB. To do this, simply enable sRGB conversion in the current Environment (more on that below).

Keep in mind that sRGB [STRIKEOUT:> Linear and Linear]> sRGB conversions must always be both enabled. Failing to enable one of them will result in horrible visuals suitable only for avant garde experimental indie games.

Parameters of HDR

HDR is found in the Environment resource. These are found most of the time inside a WorldEnvironment node, or set in a camera. There are many parameters for HDR:

ToneMapper

The ToneMapper is the heart of the algorithm. Many options for tonemappers are provided:

- Linear: Simplest tonemapper. It does its job for adjusting scene brightness, but if the differences in light are too big, it will cause colors to be too saturated.
- Log: Similar to linear, but not as extreme.
- Reinhardt: Classical tonemapper (modified so it will not desaturate as much)
- ReinhardtAutoWhite: Same as above, but uses the max scene luminance to adjust the white value.

Exposure

The same exposure parameter as in real cameras. Controls how much light enters the camera. Higher values will result in a brighter scene and lower values will result in a darker scene.

White

Maximum value of white.
Glow threshold

Determine above which value (from 0 to 1 after the scene is tonemapped), light will start bleeding.

Glow scale

Determine how much light will bleed.

Min luminance

Lower bound value of light for the scene at which the tonemapper stops working. This allows dark scenes to remain dark.

Max luminance

Upper bound value of light for the scene at which the tonemapper stops working. This allows bright scenes to remain saturated.

Exposure adjustment speed

Auto-exposure will change slowly and will take a while to adjust (like in real cameras). Bigger values means faster adjustment.

4.3.8 Using gridmaps

Introduction

Gridmaps are a simple and fast way to create 3D game levels. Think of it as a 3D version of the TileMap node. Similarly, you start with a predefined library of 3D meshes that can be put on a grid, just like if you were building a level with an unlimited amount of Lego blocks.

Collisions can also be added to the meshes, just like you would do with the tiles of a tilemap.

Creating a MeshLibrary

To begin, you need a MeshLibrary, which is a collection of meshes that can be used in the gridmap. Here are some meshes you can use to set it up.
Open a new scene and create a root node (this is important, as without the root node, it will not be able to generate the MeshLibrary!). Then, create a *MeshInstance* node:
If you don’t need to apply physics to the building blocks, that’s all you need to do. In most cases though, you will need your block to generate collisions, so let’s see how to add them.

**Collisions**

To assign a *CollisionShape* and *PhysicsBody* to the meshes, the simplest way is to do it while creating the MeshLibrary. Alternatively, you can also edit an existing MeshLibrary from within the GridMap inspector, but only CollisionShapes can be defined there and not PhysicsBody.

To give the meshes a CollisionShape, you simply add children nodes to the MeshInstance node. You would typically add the desired PhysicsBody and CollisionShape in this order:

You can adjust the order according to your needs.

**Exporting the MeshLibrary**

To export, go to *Scene > Convert To.. > MeshLibrary...* and save it as a resource.
You are now ready to use the GridMap node.

**Using the MeshLibrary in a GridMap**

Create a new scene using any node as root, then add a Gridmap node. Then, load the MeshLibrary that you just exported.
Now, you can build your own level as you see best fit. Use left click to add tiles and right click to remove them. You can adjust the floor level when you need to put meshes at specific heights.

As mentioned above, you can also define new CollisionShapes at this stage by doing the following steps:
Reminder

- Be cautious before scaling meshes if you are not using uniform meshes.
- There are many ways to make use of gridmaps, be creative!

4.4 GUI

4.4.1 GUI skinning

Oh beautiful GUI!

This tutorial is about advanced skinning of an user interface. Most games generally don’t need this, as they end up just relying on Label, TextureFrame, TextureButton and TextureProgress.

However, many types of games often need complex user interfaces, like MMOs, traditional RPGs, Simulators, Strategy, etc. These kind of interfaces are also common in some games that include editors to create content, or interfaces for network connectivity.

Godot user interface uses these kind of controls with the default theme, but they can be skinned to resemble pretty much any kind of user interface.

Theme

The GUI is skinned through the Theme resource. Theme contains all the information required to change the entire visual styling of all controls. Theme options are named, so it’s not obvious which name changes what (specialy from code), but several tools are provided. The ultimate place to look at what each theme option is for each control, which will always be more up to date than any documentation is the file scene/resources/default_theme/default_theme.cpp. The rest of this document will explain the different tools used to customize the theme.

A Theme can be applied to any control in the scene. As a result, all children and grand-children controls will use that same theme too (unless another theme is specified further down the tree). If a value is not found in a theme, it will be
searched in themes higher up in the hierarchy towards the root. If nothing was found, the default theme is used. This system allows for flexible overriding of themes in complex user interfaces.

**Theme options**

Each kind of option in a theme can be:

- **An integer constant**: A single numerical constant. Generally used to define spacing between components or alignment.
- **A Color**: A single color, with or without transparency. Colors are usually applied to fonts and icons.
- **A Texture**: A single image. Textures are not often used, but when they are they represent handles to pick or icons in a complex control (such as file dialog).
- **A Font**: Every control that uses text can be assigned the fonts used to draw strings.
- **A StyleBox**: Stylebox is a resource that defines how to draw a panel in varying sizes (more information on them later).

Every option is associated to:

- A name (the name of the option)
- A Control (the name of the control)

An example usage:

```javascript
var t = Theme.new()
t.set_color("font_color","Label",Color(1.0,1.0,1.0))

var l = Label.new()
l.set_theme(t)
```

In the example above, a new theme is created. The “font_color” option is changed and then applied to a label. As a result, the label (and all children and grand children labels) will use that color.

It is possible to override those options without using the theme directly and only for a specific control by using the override API in `Control.add_color_override()`:

```javascript
var l = Label.new()
l.add_color_override("font_color",Color(1.0,1.0,1.0))
```

In the inline help of Godot (in the script tab) you can check which theme options are overrideable, or check the `Control` class reference.

**Customizing a control**

If only a few controls need to be skinned, it is often not necessary to create a new theme. Controls offer their theme options as special kinds of properties. If checked, overriding will take place:
As can be see in the image above, theme options have little check-boxes. If checked, they can be used to override the value of the theme just for that control.

Creating a theme

The simplest way to create a theme is to edit a theme resource. Create a Theme from the resource menu, the editor will appear immediately. Following to this, save it (to, as example, mytheme.thm):
This will create an empty theme that can later be loaded and assigned to controls.

**Example: theming a button**

Take some assets (*skin_assets.zip*), go to the “theme” menu and select “Add Class Item”: 
A menu will appear prompting the type of control to create. Select “Button”:

Immediately, all button theme options will appear in the property editor, where they can be edited:
Select the “normal” stylebox and create a new “StyleBoxTexture”, then edit it. A texture stylebox basically contains a texture and the size of the margins that will not stretch when the texture is stretched. This is called “3x3” stretching:
Repeat the steps and add the other assets. There is no hover or disabled image in the example files, so use the same stylebox as in normal. Set the supplied font as the button font and change the font color to black. Soon, your button will look different and retro:
Save this theme to the .thm file. Go to the 2D editor and create a few buttons:

Now, go to the root node of the scene and locate the “theme” property, replace it by the theme that was just created. It should look like this:
Congratulations! You have created a reusable GUI Theme!

4.4.2 Custom GUI controls

So many controls...

Yet there are never enough. Creating your own custom controls that act just the way you want them is an obsession of almost every GUI programmer. Godot provides plenty of them, but they may not work exactly the way you want. Before contacting the developers with a pull-request to support diagonal scrollbars, at least it will be good to know how to create these controls easily from script.

Drawing

For drawing, it is recommended to check the Custom drawing in 2D tutorial. The same applies. Some functions are worth mentioning due to their usefulness when drawing, so they will be detailed next:
Checking control size

Unlike 2D nodes, “size” is very important with controls, as it helps to organize them in proper layouts. For this, the `Control.get_size()` method is provided. Checking it during `_draw()` is vital to ensure everything is kept in-bounds.

Checking focus

Some controls (such as buttons or text editors) might provide input focus for keyboard or joystick input. Examples of this are entering text or pressing a button. This is controlled with the `Control.set_focus_mode()` function. When drawing, and if the control supports input focus, it is always desired to show some sort of indicator (highlight, box, etc) to indicate that this is the currently focused control. To check for this status, the `Control.has_focus()` exists. Example

```python
func _draw():
    if has_focus():
        draw_selected()
    else:
        draw_normal()
```

Sizing

As mentioned before, size is very important to controls. This allows them to lay out properly, when set into grids, containers, or anchored. Controls most of the time provide a minimum size to help to properly lay them out. For example, if controls are placed vertically on top of each other using a `VBoxContainer`, the minimum size will make sure your custom control is not squished by the other controls in the container.

To provide this callback, just override `Control.get_minimum_size()`, for example:

```python
func get_minimum_size():
    return Vector2(30,30)
```

Or alternatively, set it via function:

```python
func _ready():
    set_custom_minimum_size(Vector2(30,30))
```

Input

Controls provide a few helpers to make managing input events much easier than regular nodes.

Input events

There are a few tutorials about input before this one, but it’s worth mentioning that controls have a special input method that only works when:

- The mouse pointer is over the control.
- The button was pressed over this control (control always captures input until button is released)
- Control provides keyboard/joystick focus via `Control.set_focus_mode()`.

This function is `Control.input_event()`. Simply override it in your control. No processing needs to be set.
extends Control

func _input_event(ev):
    if (ev.type==InputEvent.MOUSE_BUTTON and ev.button_index==BUTTON_LEFT and ev.pressed):
        print("Left mouse button was pressed!")

For more information about events themselves, check the InputEvent tutorial.

Notifications

Controls also have many useful notifications for which no callback exists, but can be checked with the _notification callback:

func _notification(what):
    if (what==NOTIFICATION_MOUSE_ENTER):
        pass  # mouse entered the area of this control
    elif (what==NOTIFICATION_MOUSE_EXIT):
        pass  # mouse exited the area of this control
    elif (what==NOTIFICATION_FOCUS_ENTER):
        pass  # control gained focus
    elif (what==NOTIFICATION_FOCUS_EXIT):
        pass  # control lost focus
    elif (what==NOTIFICATION_THEME_CHANGED):
        pass  # theme used to draw the control changed
              # update and redraw is recommended if using a theme
    elif (what==NOTIFICATION_VISIBILITY_CHANGED):
        pass  # control became visible/invisible
              # check new status with is_visible()
    elif (what==NOTIFICATION_RESIZED):
        pass  # control changed size, check new size
              # with get_size()
    elif (what==NOTIFICATION_MODAL_CLOSED):
        pass  # for modal popups, notification
              # that the popup was closed

4.4.3 Size and anchors

If a game was to be always run in the same device and at the same resolution, positioning controls would be a simple matter of setting the position and size of each one of them. Unfortunately, it is rarely the case.

Only TVs nowadays have a standard resolution and aspect ratio. Everything else, from computer monitors to tablets, portable consoles and mobile phones have different resolutions and aspect ratios.

There are several ways to handle this, but for now let’s just imagine that the screen resolution has changed and the controls need to be re-positioned. Some will need to follow the bottom of the screen, others the top of the screen, or maybe the right or left margins.
This is done by editing the margin properties of controls. Each control has four margins: left, right, bottom and top. By default all of them represent a distance in pixels relative to the top-left corner of the parent control or (in case there is no parent control) the viewport.
When horizontal (left,right) and/or vertical (top,bottom) anchors are changed to END, the margin values become relative to the bottom-right corner of the parent control or viewport.

Here the control is set to expand its bottom-right corner with that of the parent, so when re-sizing the parent, the control will always cover it, leaving a 20 pixel margin:
Finally, there is also a ratio option, where 0 means left, 1 means right and anything in between is interpolated.

### 4.4.4 BBCode in RichTextLabel

#### Introduction

RichTextLabel allows to display complex text markup in a control. It has a built-in API for generating the markup, but can also parse a BBCode.

Note that the BBCode tags can also be used to some extent in the XML source of the class reference.

#### Setting up

For RichTextLabel to work properly, it must be set up. This means loading the intended fonts in the relevant properties:
## Reference

<table>
<thead>
<tr>
<th>Command</th>
<th>Tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>bold</strong></td>
<td><code>[b]{text][/b]</code></td>
<td>Makes <code>{text}</code> bold.</td>
</tr>
<tr>
<td><strong>italics</strong></td>
<td><code>[i]{text}[/i]</code></td>
<td>Makes <code>{text}</code> italics.</td>
</tr>
<tr>
<td><strong>underline</strong></td>
<td><code>[u]{text}[/u]</code></td>
<td>Makes <code>{text}</code> underline.</td>
</tr>
<tr>
<td><strong>code</strong></td>
<td><code>[code]{text}[/code]</code></td>
<td>Makes <code>{text}</code> monospace.</td>
</tr>
<tr>
<td><strong>center</strong></td>
<td><code>[center]{text}[/center]</code></td>
<td>Makes <code>{text}</code> centered.</td>
</tr>
<tr>
<td><strong>right</strong></td>
<td><code>[right]{text}[/right]</code></td>
<td>Makes <code>{text}</code> right-aligned.</td>
</tr>
<tr>
<td><strong>fill</strong></td>
<td><code>[fill]{text}[/fill]</code></td>
<td>Makes <code>{text}</code> fill width.</td>
</tr>
<tr>
<td><strong>indent</strong></td>
<td><code>[indent]{text}[/indent]</code></td>
<td>Increase indent level of <code>{text}</code>.</td>
</tr>
<tr>
<td><strong>url</strong></td>
<td><code>[url]{url}[/url]</code></td>
<td>Show <code>{url}</code> as such.</td>
</tr>
<tr>
<td><strong>image</strong></td>
<td><code>[img=&lt;path&gt;]{text}[/img]</code></td>
<td>Insert image at resource <code>&lt;path&gt;</code>.</td>
</tr>
<tr>
<td><strong>font</strong></td>
<td><code>[font=&lt;path&gt;]{text}[/font]</code></td>
<td>Use custom font at <code>&lt;path&gt;</code> for <code>{text}</code>.</td>
</tr>
<tr>
<td><strong>color</strong></td>
<td><code>[color=&lt;code/name&gt;]{text}[/color]</code></td>
<td>Change <code>{text}</code> color, use # format such as #ff00ff or name.</td>
</tr>
</tbody>
</table>
Built-in color names

List of valid color names for the [color=<name>] tag:

- aqua
- black
- blue
- fuchsia
- gray
- green
- lime
- maroon
- navy
- purple
- red
- silver
- teal
- white
- yellow

Hexadecimal color codes

Any valid 6 digit hexadecimal code is supported. e.g: [color=#ffffff]white[/color]

4.5 Physics

4.5.1 Physics introduction

Our world is made of tangible matter. In our world, a piano can’t go through a wall when going into a house. It needs to use the door. Video games are often like the real world and Pac-Man can’t go through the walls of his maze (although he can teleport from the left to the right side of the screen and back).

Anyway, moving sprites around is nice but one day they have to collide properly, so let’s get to the point.

Shapes

The base collidable object in Godot’s 2D world is a Shape2D. There are many types of shapes, all of them inherit this base class:

- CircleShape2D
- RectangleShape2D
- CapsuleShape2D
- ConvexPolygonShape2D
• ConcavePolygonShape2D
• etc. (there are others check the class list).

Shapes are of type Resource, but they can be created via code easily. For example:

```python
# Create a circle
var c = CircleShape2D.new()
c.set_radius(20)

# Create a box
var b = RectangleShape2D.new()
b.set_extents(Vector2(20,10))
```

The main use for shapes is checking collision/intersection and getting resolution information. Shapes are mostly convex, (except the concavepolygon one, which is just a list of segments to check collision against). This collision check is done easily with the built-in functions like:

```python
# Check if there is a collision between two shapes, each with a transform
if b.collide(b_xform, a, a_xform):
    print("OMG Collision!")
```

Godot will return correct collision and collision info from the different calls to the Shape2D api. Collision between all shapes and transforms can be done this way, or even obtaining contact information, motion casting, etc.

**Transforming shapes**

As seen before in the collide functions, 2D shapes in Godot can be transformed by using a regular Matrix32 transform, meaning the functions can check for collisions while the shapes are scaled, moved and rotated. The only limitation to this is that shapes with curved sections (such as circle and capsule) can only be scaled uniformly. This means that circle or capsule shapes scaled in the form of an ellipse will not work properly. This is a limitation on the collision algorithm used (SAT), so make sure that your circle and capsule shapes are always scaled uniformly!

![Image showing circle and oval shapes, one working and one not working]

**When problems begin**

Even though this sounds good, reality is that collision detection alone is usually not enough in most scenarios. Many problems start arising as long as the development of the game is in progress:
Too many combinations!

Games have several dozens, hundreds, thousands! of objects that can collide and be collided. The typical approach is to test everything against everything in two for loops like this:

```python
for i in colliders:
    for j in colliders:
        if (i.collides(j)):
            do_collision_code()
```

But this scales really badly. Let’s imagine there are only 100 objects in the game. This means that 100*100=10000 collisions will need to be tested each frame. This is a lot!

Visual aid

Most of the time, creating a shape via code is not enough. We need to visually place it over a sprite, draw a collision polygon, etc. It is obvious that we need nodes to create the proper collision shapes in a scene.

Collision resolution

Imagine we solved the collision issue, we can tell easily and quickly which shapes overlap. If many of them are dynamic objects that move around, or move according to newtonian physics, solving a collision of multiple objects can be really difficult code-wise.

Introducing... Godot’s physics engine!

To solve all these problems, Godot has a physics and collision engine that is well integrated into the scene system, yet it allows different levels and layers of functionality. The built-in physics engine can be used for:

- Simple Collision Detection: See `Shape2D` API.
- Scene Kinematics: Handle shapes, collisions, broadphase, etc as nodes. See `Area2D`.
- Scene Physics: Rigid bodies and constraints as nodes. See `RigidBody2D`, and the joint nodes.

Units of measure

It is often a problem when integrating a 2D physics engine to a game that such engines are optimized to work using meters as unit of measure. Godot uses a built-in custom 2D physics engine that is designed to function properly in pixels, so all units and default values used for stabilization are tuned for this, making development more straightforward.

CollisionObject2D

`CollisionObject2D` is the (virtual) base node for everything that can be collided in 2D. Area2D, StaticBody2D, KinematicBody2D and RigidBody2D all inherit from it. This node contains a list of shapes (Shape2D) and a relative transform. This means that all collisionable objects in Godot can use multiple shapes at different transforms (offset/scale/rotation). Just remember that, as mentioned before, non-uniform scale will not work for circle and capsule shapes.
A CollisionObject2D comes in different flavors: StaticBody2D, RigidBody2D, KinematicBody2D and Area2D. However, before we dig into them, let’s have a look how to define the shape of a collision object. There are two special nodes for that.

**CollisionShape2D**

This node is a helper node, which must be created as a direct child of a CollisionObject2D-derived node: Area2D, StaticBody2D, KinematicBody2D, RigidBody2D.

The purpose of a CollisionShape2D instance is to add collision shapes to its parent class. Multiple children of type CollisionShape2D can be added to a CollisionObject2D-derived object with the effect that the parent will simply get more collision shapes. When a CollisionShape2D is edited (or added, moved, deleted) the list of shapes in the parent node is updated.

At run time, though, this node does not exist and it can’t be accessed with `get_node()`. This is because it is a helper node only for editing the collision shapes. To access the shapes created at runtime, use the CollisionObject2D API directly.

As an example, here’s the scene from the platformer, containing an Area2D (named ‘coin’) having two children: a CollisionShape2D (named ‘collision’) and a sprite (called ‘sprite’):

**CollisionPolygon2D**

This one is similar to CollisionShape2D, except that instead of assigning a shape, a polygon can be edited (drawn by the user) to determine the shape. The polygon can be convex or concave, it doesn’t matter.

Here is another scene involving a CollisionPolygon2D: A StaticBody2D has been added as a child of a sprite so that the collision object moves together with the sprite. In turn, the CollisionPolygon is a child of StaticBody2D, meaning it adds collision shapes to it.
The CollisionPolygon2D will decompose the user-defined polygon into convex shapes (shapes can only be convex, remember?) before adding them to the CollisionObject2D. The following image shows such a decomposition:

**Triggers**

A CollisionShape2D or CollisionPolygon2D can be set as a trigger by setting the boolean flag with the same name. When used in a RigidBody2D or KinematicBody2D, "trigger" shapes take part in collision detection but are unaffected by physics (they don’t block movement etc). Defining a collision shape as a trigger is mostly useful in two situations:

- Collisions for a specific shape shall be disabled.
• An Area2D shall send `body_enter` and `body_exit` signals when the trigger shape enters it (useful in several situations).

**StaticBody2D**

The simplest node in the physics engine is the `StaticBody2D`. This node takes part in collision detection. However, it does not move or rotate after a collision, so physics do not influence them. The node is static. Other nodes can collide against it and will be influenced accordingly.

The platformer example uses StaticBody2D objects for the floors and walls. These are the static parts of a level and shall stay right where they are even if the player jumps onto them.

**KinematicBody2D**

Similar to StaticBody2D, objects of type `KinematicBody2D` are not affected by physics (although they take part in collision detection, of course). However, KinematicBody2D are not static but can be moved via code or an animation. They have two main uses:

• **Simulated Motion**: When these bodies are moved manually, either from code or from an `AnimationPlayer` (with process mode set to fixed!), the physics will automatically compute an estimate of their linear and angular velocity. This makes them very useful for moving platforms or other AnimationPlayer-controlled objects (like a door, a bridge that opens, etc). As an example, the 2d/platformer demo uses them for moving platforms.

• **Kinematic Characters**: KinematicBody2D also has an API for moving objects (the `move()` function) while performing collision tests. This makes them really useful to implement characters that collide against a world, but that don’t require advanced physics. There is a tutorial about it [Kinematic Character (2D)](https://docs.godotengine.org/en/4.5/physics.html#kinematic-character-2d).

**RigidBody2D**

This type of body simulates Newtonian physics. It has mass, friction, bounce, and the (0,0) coordinates simulate the center of mass. When real physics are needed, `RigidBody2D` is the node to use. The motion of this body is affected by gravity and/or other bodies.

Rigid bodies are usually active all the time, but when they end up in resting position and don’t move for a while, they are put to sleep until something else wakes them up. This saves an enormous amount of CPU.

RigidBody2D nodes update their transform constantly, as it is generated by the simulation from a position, linear velocity and angular velocity. As a result, [STRIKEOUT: this node can’t be scaled]. Scaling the children nodes should work fine though.

A RigidBody2D has a `Mode` flag to change its behavior (something which is very common in games). It can behave like a rigid body, a character (a rigid body without the ability to rotate so that it is always upright), a kinematic body or a static body. This flag can be changed dynamically according to the current situation. For example, an enemy frozen by an ice beam becomes a static body.

The best way to interact with a RigidBody2D is during the force integration callback by overriding the function

```godot
func _integrate_forces(state):
    [use state to change the object]
```

The “state” parameter is of type `Physics2DDirectBodyState`. Please do not use the state object outside the callback as it will result in an error.

During the evaluation of the aforementioned function, the physics engine synchronizes state with the scene and allows full modification of the object’s parameters. Since physics may run in its own thread, parameter changes outside that callback will not take place until the next frame.
Contact reporting

In general, RigidBody2D will not keep track of the contacts, because this can require a huge amount of memory if thousands of rigid bodies are in the scene. To get contacts reported, simply increase the amount of the “contacts reported” property from zero to a meaningful value (depending on how many you are expecting to get). The contacts can be later obtained via the `Physics2DDirectBodyState.get_contact_count()` and related functions.

Contact monitoring via signals is also available (signals similar to the ones in Area2D, described below) via a boolean property.

Area2D

Areas in Godot physics have three main roles:

1. Override the space parameters for objects entering them (i.e. gravity, gravity direction, gravity type, density, etc).
2. Monitor when rigid or kinematic bodies enter or exit the area.
3. Monitor other areas (this is the simplest way to get overlap test)

The second function is the most common. For it to work, the “monitoring” property must be enabled (it is by default). There are two types of signals emitted by this node:

```
# Simple, high level notification
body_enter(body:PhysicsBody2D)
body_exit(body:PhysicsBody2D)
area_enter(area:Area2D)
area_exit(body:Area2D)

# Low level shape-based notification
# Notifies which shape specifically in both the body and area are in contact
body_enter_shape(body_id:int,body:PhysicsBody2D,body_shape_index:int,area_shape_index:int)
body_exit_shape(body_id:int,body:PhysicsBody2D,body_shape_index:int,area_shape_index:int)
area_enter_shape(area_id:int,area:Area2D,area_shape_index:int,self_shape_index:int)
area_exit_shape(area_id:int,area:Area2D,area_shape_index:int,self_shape_index:int)
```

By default, areas also receive mouse/touchscreen input, providing a lower-level way than controls to implement this kind of input in a game. Bodies support this but it’s disabled by default.

In case of overlap, who receives collision information?

Remember that not every combination of two bodies can “report” contacts. Static bodies are passive and will not report contacts when hit. Kinematic Bodies will report contacts but only against Rigid/Character bodies. Area2D will report overlap (not detailed contacts) with bodies and with other areas. The following table should make it more visual:

<table>
<thead>
<tr>
<th>Type</th>
<th>RigidBody</th>
<th>CharacterBody</th>
<th>KinematicBody</th>
<th>StaticBody</th>
<th>Area</th>
</tr>
</thead>
<tbody>
<tr>
<td>RigidBody</td>
<td>Both</td>
<td>Both</td>
<td>Both</td>
<td>RigidBody</td>
<td>Area</td>
</tr>
<tr>
<td>CharacterBody</td>
<td>Both</td>
<td>Both</td>
<td>Both</td>
<td>CharacterBody</td>
<td>Area</td>
</tr>
<tr>
<td>KinematicBody</td>
<td>Both</td>
<td>Both</td>
<td>None</td>
<td>None</td>
<td>Area</td>
</tr>
<tr>
<td>StaticBody</td>
<td>RigidBody</td>
<td>CharacterBody</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>Area</td>
<td>Area</td>
<td>Area</td>
<td>Area</td>
<td>None</td>
<td>Both</td>
</tr>
</tbody>
</table>
Physics global variables

A few global variables can be tweaked in the project settings for adjusting how 2D physics works:

![Physics2D Settings](image)

Leaving them alone is best (except for the gravity, that needs to be adjusted in most games), but there is one specific parameter that might need tweaking which is the “cell_size”. Godot 2D physics engine used by default a space hashing algorithm that divides space in cells to compute close collision pairs more efficiently.

If a game uses several colliders that are really small and occupy a small portion of the screen, it might be necessary to shrink that value (always to a power of 2) to improve efficiency. Likewise if a game uses few large colliders that span a huge map (of several screens of size), increasing that value a bit might help save resources.

Fixed process callback

The physics engine may spawn multiple threads to improve performance, so it can use up to a full frame to process physics. Because of this, when accessing physics variables such as position, linear velocity, etc. they might not be representative of what is going on in the current frame.

To solve this, Godot has a fixed process callback, which is like process but it’s called once per physics frame (by default 60 times per second). During this time, the physics engine is in synchronization state and can be accessed directly and without delays.

To enable a fixed process callback, use the set_fixed_process() function, example:

```godot
extends KinematicBody2D

func _fixed_process(delta):
    move(direction * delta)

func _ready():
    set_fixed_process(true)
```

Casting rays and motion queries

It is very often desired to “explore” the world around from our code. Throwing rays is the most common way to do it. The simplest way to do this is by using the RayCast2D node, which will throw a ray every frame and record the intersection.

At the moment there isn’t a high level API for this, so the physics server must be used directly. For this, the Physics2DDirectspaceState class must be used. To obtain it, the following steps must be taken:

1. It must be used inside the _fixed_process() callback, or at _integrate_forces()
2. The 2D RIDs for the space and physics server must be obtained.

The following code should work:
func _fixed_process(delta):
    var space = get_world_2d().get_space()
    var space_state = Physics2DServer.space_get_direct_state(space)

Enjoy doing space queries!

### 4.5.2 Ray-casting

#### Introduction

One of the most common tasks in game development is casting a ray (or custom shaped object) and checking what it hits. This enables complex behaviors, AI, etc. to take place. This tutorial will explain how to do this in 2D and 3D.

Godot stores all the low level game information in servers, while the scene is just a frontend. As such, ray casting is generally a lower-level task. For simple raycasts, node such as `RayCast` and `RayCast2D` will work, as they will return every frame what the result of a raycast is.

Many times, though, ray-casting needs to be a more interactive process so a way to do this by code must exist.

#### Space

In the physics world, Godot stores all the low level collision and physics information in a *space*. The current 2d space (for 2D Physics) can be obtained by calling `CanvasItem.get_world_2d().get_space()`. For 3D, it’s `Spatial.get_world().get_space()`.

The resulting space `RID` can be used in `PhysicsServer` and `Physics2DServer` respectively for 3D and 2D.

#### Accessing space

Godot physics runs by default in the same thread as game logic, but may be set to run on a separate thread to work more efficiently. Due to this, the only time accessing space is safe is during the `Node._fixed_process()` callback. Accessing it from outside this function may result in an error due to space being *locked*.

To perform queries into physics space, the `Physics2DDirectSpaceState` and `PhysicsDirectSpaceState` must be used.

In code, for 2D spacestate, this code must be used:

```godot
func _fixed_process(delta):
    var space_rid = get_world_2d().get_space()
    var space_state = Physics2DServer.space_get_direct_state(space_rid)
```

Of course, there is a simpler shortcut:

```godot
func _fixed_process(delta):
    var space_state = get_world_2d().get_direct_space_state()
```

For 3D:

```godot
func _fixed_process(delta):
    var space_state = get_world().get_direct_space_state()
```
Raycast query

For performing a 2D raycast query, the method `Physics2DDirectSpaceState.intersect_ray()` must be used, for example:

```python
func _fixed_process(delta):
    var space_state = get_world().get_direct_space_state()
    # use global coordinates, not local to node
    var result = space_state.intersect_ray( Vector2(0,0), Vector2(50,100) )
```

Result is a dictionary. If the ray didn’t hit anything, the dictionary will be empty. If it did hit something it will contain collision information:

```python
if (not result.empty()):
    print("Hit at point: ", result.position)
```

The collision result dictionary, when something hit, has this format:

```json
{
    position:Vector2 # point in world space for collision
    normal:Vector2 # normal in world space for collision
    collider:Object # Object collided or null (if unassociated)
    collider_id:ObjectID # Object it collided against
    rid:RID # RID it collided against
    shape:int # shape index of collider
    metadata:Variant() # metadata of collider
}
```

# in case of 3D, Vector3 is returned.

Collision exceptions

It is a very common case to attempt casting a ray from a character or another game scene to try to infer properties of the world around it. The problem with this is that the same character has a collider, so the ray can never leave the origin (it will keep hitting its own collider), as evidenced in the following image.
To avoid self-intersection, the `intersect_ray()` function can take an optional third parameter which is an array of exceptions. This is an example of how to use it from a KinematicBody2D or any other collisionobject based node:

```godot
extends KinematicBody2D
func _fixed_process(delta):
    var space_state = get_world().get_direct_space_state()
    var result = space_state.intersect_ray( get_global_pos(), enemy_pos, [ self ] )
```

The extra argument is a list of exceptions, can be objects or RIDs.

### 3D ray casting from screen

Casting a ray from screen to 3D physics space is useful for object picking. There is not much of a need to do this because `CollisionObject` has an “input_event” signal that will let you know when it was clicked, but in case there is any desire to do it manually, here’s how.

To cast a ray from the screen, the `Camera` node is needed. Camera can be in two projection modes, perspective and orthogonal. Because of this, both the ray origin and direction must be obtained. (origin changes in orthogonal, while direction changes in perspective):
To obtain it using a camera, the following code can be used:

```plaintext
const ray_length = 1000

func _input(ev):
    if ev.type==InputEvent.MOUSE_BUTTON and ev.pressed and ev.button_index==1:
        var camera = get_node("camera")
        var from = camera.project_ray_origin(ev.pos)
        var to = from + camera.project_ray_normal(ev.pos) * ray_length
```

Of course, remember that during `_input()`, space may be locked, so save your query for `_fixed_process()`.

4.5.3 Kinematic Character (2D)

Introduction

Yes, the name sounds strange. “Kinematic Character”. What is that? The reason is that when physics engines came out, they were called “Dynamics” engines (because they dealt mainly with collision responses). Many attempts were made to create a character controller using the dynamics engines but it wasn’t as easy as it seems. Godot has one of the best implementations of dynamic character controller you can find (as it can be seen in the 2d/platformer demo), but using it requires a considerable level of skill and understanding of physics engines (or a lot of patience with trial and error).

Some physics engines such as Havok seem to swear by dynamic character controllers as the best alternative, while others (PhysX) would rather promote the Kinematic one.

So, what is really the difference?:

- A **dynamic character controller** uses a rigid body with infinite inertial tensor. Basically, it’s a rigid body that can’t rotate. Physics engines always let objects collide, then solve their collisions all together. This makes dynamic character controllers able to interact with other physics objects seamlessly (as seen in the platformer demo), however these interactions are not always predictable. Collisions also can take more than one frame to be solved, so a few collisions may seem to displace a tiny bit. Those problems can be fixed, but require a certain amount of skill.
• A kinematic character controller is assumed to always begin in a non-colliding state, and will always move to a non-colliding state. If it starts in a colliding state, it will try to free itself (like rigid bodies do) but this is the exception, not the rule. This makes their control and motion a lot more predictable and easier to program. However, as a downside, they can’t directly interact with other physics objects (unless done by hand in code).

This short tutorial will focus on the kinematic character controller. Basically, the oldschool way of handling collisions (which is not necessarily simpler under the hood, but well hidden and presented as a nice and simple API).

Fixed process

To manage the logic of a kinematic body or character, it is always advised to use fixed process, which is called the same amount of times per second, always. This makes physics and motion calculation work in a more predictable way than using regular process, which might have spikes or lose precision if the frame rate is too high or too low.

```python
extends KinematicBody2D
func _fixed_process(delta):
    pass
func _ready():
    set_fixed_process(true)
```

Scene setup

To have something to test, here’s the scene (from the tilemap tutorial): kbscene.zip. We’ll be creating a new scene for the character. Use the robot sprite and create a scene like this:

Let’s add a circular collision shape to the collision body, create a new CircleShape2D in the shape property of CollisionShape2D. Set the radius to 30:
Note: As mentioned before in the physics tutorial, the physics engine can’t handle scale on most types of shapes (only collision polygons, planes and segments work), so always change the parameters (such as radius) of the shape instead of scaling it. The same is also true for the kinematic/rigid/static bodies themselves, as their scale affect the shape scale.

Now create a script for the character, the one used as an example above should work as a base.

Finally, instance that character scene in the tilemap, and make the map scene the main one, so it runs when pressing play.

Moving the Kinematic character

Go back to the character scene, and open the script, the magic begins now! Kinematic body will do nothing by default, but it has a really useful function called `KinematicBody2D.move()`. This function takes a `Vector2` as an argument, and tries to apply that motion to the kinematic body. If a collision happens, it stops right at the moment of the collision.
So, let’s move our sprite downwards until it hits the floor:

```godot
extends KinematicBody2D

func _fixed_process(delta):
    move( Vector2(0,1) )  # move down 1 pixel per physics frame

func _ready():
    set_fixed_process(true)
```

The result is that the character will move, but stop right when hitting the floor. Pretty cool, huh?

The next step will be adding gravity to the mix, this way it behaves a little more like an actual game character:

```godot
extends KinematicBody2D

const GRAVITY = 200.0
var velocity = Vector2()

func _fixed_process(delta):
    velocity.y += delta * GRAVITY
    var motion = velocity * delta
    move( motion )

func _ready():
    set_fixed_process(true)
```

Now the character falls smoothly. Let’s make it walk to the sides, left and right when touching the directional keys. Remember that the values being used (for speed at least) is pixels/second.

This adds simple walking support by pressing left and right:

```godot
extends KinematicBody2D

const GRAVITY = 200.0
const WALK_SPEED = 200

var velocity = Vector2()

func _fixed_process(delta):
    velocity.y += delta * GRAVITY
    if (Input.is_action_pressed("ui_left")):
        velocity.x = -WALK_SPEED
    elif (Input.is_action_pressed("ui_right")):
        velocity.x = WALK_SPEED
    else:
        velocity.x = 0
    var motion = velocity * delta
    move(motion)

func _ready():
    set_fixed_process(true)
```

And give it a try.
Problem?

And... it doesn’t work very well. If you go to the left against a wall, it gets stuck unless you release the arrow key. Once it is on the floor, it also gets stuck and it won’t walk. What is going on??

The answer is, what it seems like it should be simple, it isn’t that simple in reality. If the motion can’t be completed, the character will stop moving. It’s as simple as that. This diagram should illustrate better what is going on:

![Diagram showing the problem and solution]

Basically, the desired motion vector will never complete because it hits the floor and the wall too early in the motion trajectory and that makes it stop there. Remember that even though the character is on the floor, the gravity is always turning the motion vector downwards.

Solution!

The solution? This situation is solved by “sliding” by the collision normal. KinematicBody2D provides two useful functions:

- `KinematicBody2D.is_colliding()`
- `KinematicBody2D.get_collision_normal()`

So what we want to do is this:
When colliding, the function `move()` returns the “remainder” of the motion vector. That means, if the motion vector is 40 pixels, but collision happened at 10 pixels, the same vector but 30 pixels long is returned.

The correct way to solve the motion is, then, to slide by the normal this way:

```python
func _fixed_process(delta):
    velocity.y += delta * GRAVITY
    if (Input.is_action_pressed("ui_left")):
        velocity.x = - WALK_SPEED
    elif (Input.is_action_pressed("ui_right")):
        velocity.x = WALK_SPEED
    else:
        velocity.x = 0
    var motion = velocity * delta
    motion = move(motion)
    if (is_colliding()):
        var n = get_collision_normal()
        motion = n.slide(motion)
        velocity = n.slide(velocity)
        move(motion)

func _ready():
    set_fixed_process(true)
```

Note that not only the motion has been modified but also the velocity. This makes sense as it helps keep the new direction too.

The normal can also be used to detect that the character is on floor, by checking the angle. If the normal points up (or at least, within a certain threshold), the character can be determined to be there.
A more complete demo can be found in the demo zip distributed with the engine, or in the https://github.com/godotengine/godot-demo-projects/tree/2.1/2d/kinematic_char.

4.6 Viewports

4.6.1 Viewports

Introduction

Godot has a small but very useful feature called viewports. Viewports are, as they name implies, rectangles where the world is drawn. They have three main uses, but can flexibly adapted to a lot more. All this is done via the Viewport node.

The main uses in question are:

- **Scene Root**: The root of the active scene is always a Viewport. This is what displays the scenes created by the user. (You should know this by having read previous tutorials!)
- **Sub-Viewports**: These can be created when a Viewport is a child of a Control.
- **Render Targets**: Viewports can be set to “RenderTarget” mode. This means that the viewport is not directly visible, but its contents can be accessed via a Texture.

Input

Viewports are also responsible of delivering properly adjusted and scaled input events to all its children nodes. Both the root viewport and sub-viewports do this automatically, but render targets do not. Because of this, the user must do it manually via the Viewport.input() function if needed.

Listener

Godot supports 3D sound (in both 2D and 3D nodes), more on this can be found in another tutorial (one day..). For this type of sound to be audible, the viewport needs to be enabled as a listener (for 2D or 3D). If you are using a custom viewport to display your world, don’t forget to enable this!

Cameras (2D & 3D)

When using a 2D or 3D Camera / Camera2D, cameras will always display on the closest parent viewport (going towards the root). For example, in the following hierarchy:

- Viewport
  - Camera

Camera will display on the parent viewport, but in the following one:

- Camera
  - Viewport
It will not (or may display in the root viewport if this is a subscene).

There can be only one active camera per viewport, so if there is more than one, make sure that the desired one has the “current” property set, or make it the current camera by calling:

```plaintext
camera.make_current()
```

**Scale & stretching**

Viewports have a “rect” property. X and Y are often not used (only the root viewport really uses them), while WIDTH AND HEIGHT represent the size of the viewport in pixels. For Sub-Viewports, these values are overridden by the ones from the parent control, but for render targets this sets their resolution.

It is also possible to scale the 2D content and make it believe the viewport resolution is other than the one specified in the rect, by calling:

```plaintext
viewport.set_size_override(w, h)  # custom size for 2D
viewport.set_size_override_stretch(true/false)  # enable stretch for custom size
```

The root viewport uses this for the stretch options in the project settings.

**Worlds**

For 3D, a Viewport will contain a **World**. This is basically the universe that links physics and rendering together. Spatial-base nodes will register using the World of the closest viewport. By default, newly created viewports do not contain a World but use the same as a parent viewport (root viewport does contain one though, which is the one objects are rendered to by default). A world can be set in a viewport using the “world” property, and that will separate all children nodes of that viewport from interacting with the parent viewport world. This is specially useful in scenarios where, for example, you might want to show a separate character in 3D imposed over the game (like in Starcraft).

As a helper for situations where you want to create viewports that display single objects and don’t want to create a world, viewport has the option to use its own World. This is very useful when you want to instance 3D characters or objects in the 2D world.

For 2D, each Viewport always contains its own **World2D**. This suffices in most cases, but in case sharing them may be desired, it is possible to do so by calling the viewport API manually.

**Capture**

It is possible to query a capture of the viewport contents. For the root viewport this is effectively a screen capture. This is done with the following API:

```plaintext
# queues a screen capture, will not happen immediately
viewport.queue_screen_capture()
```

After a frame or two (check `_process()`), the capture will be ready, get it back by using:

```plaintext
var capture = viewport.get_screen_capture()
```

If the returned image is empty, capture still didn’t happen, wait a little more, as this API is asynchronous.
Sub-viewport

If the viewport is a child of a control, it will become active and display anything it has inside. The layout is something like this:

- Control
  - Viewport

The viewport will cover the area of its parent control completely.

Render target

To set as a render target, just toggle the “render target” property of the viewport to enabled. Note that whatever is inside will not be visible in the scene editor. To display the contents, the render target texture must be used. This can be requested via code using (for example):

```javascript
var rtt = viewport.get_render_target_texture()
sprite.set_texture(rtt)
```

By default, re-rendering of the render target happens when the render target texture has been drawn in a frame. If visible, it will be rendered, otherwise it will not. This behavior can be changed to manual rendering (once), or always render, no matter if visible or not.

A few classes are created to make this easier in most common cases inside the editor:

- `ViewportSprite` (for 2D).
- `ViewportQuad` (for 3D).
• ViewportFrame (for GUI).

TODO: Review the doc, ViewportQuad and ViewportFrame don’t exist in 2.0.

Make sure to check the viewport demos! Viewport folder in the demos archive available to download, or https://github.com/godotengine/godot-demo-projects/tree/master/viewport

4.6.2 Multiple resolutions

Base resolution

A base screen resolution for the project can be specified in the project settings.

However, what it does is not completely obvious. When running on PC, the engine will attempt to set this resolution (or use something smaller if it fails). On mobile, consoles or devices with a fixed resolution or full screen rendering, this resolution will be ignored and the native resolution will be used instead. To compensate for this, Godot offers many ways to control how the screen will resize and stretch to different screen sizes.

Resizing

There are several types of devices, with several types of screens, which in turn have different pixel density and resolutions. Handling all of them can be a lot of work, so Godot tries to make the developer’s life a little easier. The Viewport node has several functions to handle resizing, and the root node of the scene tree is always a viewport (scenes loaded are instanced as a child of it, and it can always be accessed by calling get_tree().get_root() or get_node("/root").

In any case, while changing the root Viewport params is probably the most flexible way to deal with the problem, it can be a lot of work, code and guessing, so Godot provides a simple set of parameters in the project settings to handle multiple resolutions.

Stretch settings

Stretch settings are located in the project settings, it’s just a bunch of configuration variables that provide several options:
Stretch mode

- **Disabled**: The first is the stretch mode. By default this is disabled, which means no stretching happens (the bigger the screen or window, the bigger the resolution, always matching pixels 1:1).

- **2D**: In this mode, the resolution specified in display/width and display/height in the project settings will be stretched to cover the whole screen. This means that 3D will be unaffected (will just render to higher-res) and 2D will also be rendered at higher-res, just enlarged.

- **Viewport**: Viewport scaling is different, the root Viewport is set as a render target, and still renders precisely to the resolution specified in the display/ section of the project settings. Finally, this viewport is copied and scaled to fit the screen. This mode is useful when working with pixel-precise games, or just for the sake of rendering to a lower resolution for improving performance.

![disabled 2d viewport](image)

Stretch aspect

- **Ignore**: Ignore the aspect ratio when stretching the screen. This means that the original resolution will be stretched to fit the new one, even if it’s wider or narrower.

- **Keep**: Keep aspect ratio when stretching the screen. This means that the original resolution will be kept when fitting the new one, and black bars will be added to the sides or the top/bottom of the screen.

- **Keep Width**: Keep aspect ratio when stretching the screen, but if the resulting screen is taller than the specified resolution, it will be stretched vertically (and more vertical resolution will be reported in the viewport, proportionally). This is usually the best option for creating GUIs or HUDs that scale, so some controls can be anchored to the bottom (Size and anchors).

- **Keep Height**: Keep aspect ratio when stretching the screen, but if the resulting screen is wider than the specified resolution, it will be stretched horizontally (and more horizontal resolution will be reported in the viewport, proportionally). This is usually the best option for 2D games that scroll horizontally (like runners or platformers).

4.7 Inputs

4.7.1 InputEvent

What is it?

Managing input is usually complex, no matter the OS or platform. To ease this a little, a special built-in type is provided, `InputEvent`. This datatype can be configured to contain several types of input events. Input Events travel through the engine and can be received in multiple locations, depending on the purpose.

How does it work?

Every input event is originated from the user/player (though it’s possible to generate an InputEvent and feed them back to the engine, which is useful for gestures). The OS object for each platform will read events from the device, then
feed them to MainLoop. As `SceneTree` is the default MainLoop implementation, events are fed to it. Godot provides a function to get the current SceneTree object: `get_tree()`.

But `SceneTree` does not know what to do with the event, so it will give it to the viewports, starting by the “root” `Viewport` (the first node of the scene tree). Viewport does quite a lot of stuff with the received input, in order:

1. First of all, the standard `_input` function will be called in any node with input processing enabled (enable with `Node.set_process_input()` and override `Node._input()`). If any function consumes the event, it can call `SceneTree.set_input_as_handled()`, and the event will not spread any more. This ensures that you can filter all events of interest, even before the GUI. For gameplay input, the `_unhandled_input()` is generally a better fit, because it allows the GUI to intercept the events.

2. Second, it will try to feed the input to the GUI, and see if any control can receive it. If so, the `Control` will be called via the virtual function `Control._input_event()` and the signal “input_event” will be emitted (this function is re-implementable by script by inheriting from it). If the control wants to “consume” the event, it will call `Control.accept_event()` and the event will not spread any more.

3. If so far no one consumed the event, the unhandled input callback will be called (enable with
Node.set_process_unhandled_input() and override Node._unhandled_input()). If any function consumes the event, it can call SceneTree.set_input_as_handled(), and the event will not spread any more. The unhandled input callback is ideal for full-screen gameplay events, so they are not received when a GUI is active.

4. If no one wanted the event so far, and a Camera is assigned to the Viewport, a ray to the physics world (in the ray direction from the click) will be cast. If this ray hits an object, it will call the CollisionObject.input_event() function in the relevant physics object (bodies receive this callback by default, but areas do not. This can be configured through Area properties).

5. Finally, if the event was unhandled, it will be passed to the next Viewport in the tree, otherwise it will be ignored.

**Anatomy of an InputEvent**

InputEvent is just a base built-in type, it does not represent anything and only contains some basic information, such as event ID (which is increased for each event), device index, etc.

InputEvent has a “type” member. By assigning it, it can become different types of input event. Every type of InputEvent has different properties, according to its role.

Example of changing event type.

```cpp
# create event
var ev = InputEvent()
# set type index
ev.type = InputEvent.MOUSE_BUTTON
# button_index is only available for the above type
ev.button_index = BUTTON_LEFT
```

There are several types of InputEvent, described in the table below:

<table>
<thead>
<tr>
<th>Event</th>
<th>Type Index</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InputEvent</td>
<td>NONE</td>
<td>Empty Input Event.</td>
</tr>
<tr>
<td>InputKeyEvent</td>
<td>KEY</td>
<td>Contains a scancode and unicode value, as well as modifiers.</td>
</tr>
<tr>
<td>InputEventMouseButton</td>
<td>MOUSE_BUTTON</td>
<td>Contains click information, such as button, modifiers, etc.</td>
</tr>
<tr>
<td>InputEventMouseMotion</td>
<td>MOUSE_MOTION</td>
<td>Contains motion information, such as relative, absolute positions and speed.</td>
</tr>
<tr>
<td>InputEventJoyStickMotion</td>
<td>JOY-STICK_MOTION</td>
<td>Contains Joystick/Joypad analog axis information.</td>
</tr>
<tr>
<td>InputEventScreenTouch</td>
<td>SCREEN_TOUCH</td>
<td>Contains multi-touch press/release information. (only available on mobile devices)</td>
</tr>
<tr>
<td>InputEventScreenDrag</td>
<td>SCREEN_DRAG</td>
<td>Contains multi-touch drag information. (only available on mobile devices)</td>
</tr>
<tr>
<td>InputEventAction</td>
<td>SCREEN_ACTION</td>
<td>Contains a generic action. These events are often generated by the programmer as feedback. (more on this below)</td>
</tr>
</tbody>
</table>

**Actions**

An InputEvent may or may not represent a pre-defined action. Actions are useful because they abstract the input device when programming the game logic. This allows for:

- The same code to work on different devices with different inputs (e.g., keyboard on PC, Joypad on console).
• Input to be reconfigured at run-time.

Actions can be created from the Project Settings menu in the Actions tab. Read Input actions setup for an explanation on how the action editor works.

Any event has the methods InputEvent.is_action(), InputEvent.is_pressed() and InputEvent.

Alternatively, it may be desired to supply the game back with an action from the game code (a good example of this is detecting gestures). The Input singleton has a method for this: Input.parse_input_event(). You would normally use it like this:

```python
var ev = InputEvent()
ev.type = InputEvent.ACTION
# set as move_left, pressed
ev.set_as_action("move_left", true)
# feedback
Input.parse_input_event(ev)
```

**InputMap**

Customizing and re-mapping input from code is often desired. If your whole workflow depends on actions, the InputMap singleton is ideal for reassigning or creating different actions at run-time. This singleton is not saved (must be modified manually) and its state is run from the project settings (engine.cfg). So any dynamic system of this type needs to store settings in the way the programmer best sees fit.

### 4.7.2 Mouse and input coordinates

**About**

The reason for this small tutorial is to clear up many common mistakes about input coordinates, obtaining mouse position and screen resolution, etc.

**Hardware display coordinates**

Using hardware coordinates makes sense in the case of writing complex UIs meant to run on PC, such as editors, MMOs, tools, etc. Yet, it does not make as much sense outside of that scope.

**Viewport display coordinates**

Godot uses viewports to display content, and viewports can be scaled by several options (see Multiple resolutions tutorial). Use, then, the functions in nodes to obtain the mouse coordinates and viewport size, for example:

```python
func _input(ev):
    # Mouse in viewport coordinates
    if (ev.type==InputEvent.MOUSE_BUTTON):
        print("Mouse Click/Unclick at: ",ev.pos)
    elif (ev.type==InputEvent.MOUSE_MOTION):
        print("Mouse Motion at: ",ev.pos)

    # Print the size of the viewport
    print("Viewport Resolution is: ",get_viewport_rect().size)
```
func _ready():
    set_process_input(true)

Alternatively it’s possible to ask the viewport for the mouse position:

get_viewport().get_mouse_pos()

4.8 Animation

4.8.1 Introduction to the 2D animation features

Overview

The AnimationPlayer node allows you to create anything from simple to complex animations. In this guide you learn to:

- Work with the Animation Panel
- Animate any property of any node
- Create a simple animation
- Call functions with the powerful Call Function Tracks

In Godot you can animate anything you find in the Inspector. Animations are changes to properties over time. This means you can animate anything visual in nature: sprites, UI elements, particles, the visibility, and color of textures. But not only. You can also control gameplay related values and call any function.

Create an AnimationPlayer node

To use the animation tools we first have to create an AnimationPlayer node.

The AnimationPlayer node type is the data container for your animations. One AnimationPlayer node can hold multiple animations, that can automatically transition to one another.

Click on the AnimationPlayer node in the Node tab to open the Animation Panel at the bottom of the viewport.

It consists of five parts:

- Animation controls (i.e. add, load, save, and delete animations)
- The tracks listing
- The timeline with keyframes
- The track modifiers and keyframe editor (when enabled)
- The timeline and track controls, where you can zoom the timeline and edit tracks for example.

See the animation panel reference below for details.
Fig. 4.1: The AnimationPlayer node

Fig. 4.2: The animation panel position
Computer animation relies on keyframes

On the computer animations rely on keyframes. A keyframe defines the value of a property at a certain point in time. White and blue diamond shapes represent keyframes in the timeline. The engine interpolates values between keyframes, resulting in a gradual change in values over time. The timeline lets you insert keyframes and change their timing. It also defines how long the animation is. Each line of the Animation Panel is an animation track. Normal and Transform tracks reference node properties. Their name or id is a path to the node.

Tip: If you animate the wrong property, you can edit a track’s path anytime. Double click on it and type the new path.

Play the animation using the “Play from beginning” button (Default shortcut: Shift+D) to see the changes instantly.

Tutorial: Creating a simple animation

Scene setup

For this tutorial, we’ll going to create an AnimationPlayer node and a sprite node as a AnimationPlayer node child. The sprite holds an image texture and we animate that sprite to move between two points on the screen. As a starting point, move the sprite to a left position on the screen.
Fig. 4.5: Two keyframes are all it takes to obtain a smooth motion

Fig. 4.6: The timeline in the animation panel

Fig. 4.7: Example of Normal animation tracks
Tip: Adding animated nodes as children to the AnimationPlayer node is not required, but it is a nice way of distinguishing animated nodes from non-animated nodes in the Scene Tree.

Select the AnimationPlayer node and click on “Add animation” (fig. 4.9) in the animation tab to add a new animation. Enter a name for the animation in the dialog box.
Adding a track

To add a new track for our sprite, select it and take a look in the toolbar:

![Fig. 4.10: Convenience buttons](image)

These switches and buttons allow you to add keyframes for the selected node’s location, rotation, and scale respectively.

Deselect rotation, because we are only interested in the location of our sprite for this tutorial and click on the key button.

As we don’t have a track already set up for the transform/location property, Godot asks, wether it should set it up for us. Click on “Create”.

This creates a new track and our very first keyframe at the beginning of the timeline:

![Fig. 4.11: The sprite track](image)

The track name consists of a Node Path, followed by a colon, followed by a reference to its property, that we would like to modify.

In our example, the path is `AnimationPlayer/Sprite` and the property is `transform/pos`.

The path always starts at the AnimationPlayer node’s parent (so paths always have to include the AnimationPlayer node itself).

---

**Note:** Don’t worry if you change the names of nodes in the Scene Tree, that you already have tracks for. Godot automatically updates the paths in the tracks.

---

The second keyframe

Now we need to set the destination where our sprite should be headed and how much time it takes to get there.

Let’s say, we want it to take 2 seconds to go to the other point. By default the animation is set to last only 1 second, so change this in the timeline controls animation panel’s lower panel to 2.

![Fig. 4.12: Animation length](image)

Click on the timeline header near the 2 second mark and move the sprite to the target destination on the right side. Again, click the key button in the toolbar. This creates our second keyframe.
Run the animation

Click on the “Play from beginning” ( ) button.

Yay! Our animation runs:

Fig. 4.13: The animation

Back and forth

As you can see, the “loop” button is enabled by default and our animation loops. Godot has an additional feature here. Like said before, Godot always calculates the frames between two keyframes. In a loop, the first keyframe is also the last keyframe, if no keyframe is specified at the end.

Fig. 4.14: Animation loop

If you set the animation length to 4 seconds now, the animation moves back and forth.

Track settings

Each track has a settings panel at the end, where you can set the update rate and the track interpolation.

Fig. 4.15: Track settings

The update rate of a track tells Godot when to update the property values. This can be:

- Continuous: Update the property on each frame
- Discrete: Only update the property on keyframes
- Trigger: Only update the property on keyframes or triggers

In normal animations, you usually use “Continuous”. The other types are used to script complex animations.

The interpolation tells Godot how to calculate the frame values between the keyframes. These interpolation modes are supported:

- Nearest: Set the nearest keyframe value
- Linear: Set the value based on a linear function calculation between the two keyframes
- Cubic: Set the value based on a curved function calculation between the two keyframes
Cubic interpolation leads to a more natural movement, where the animation is slower at a keyframe and faster between keyframes. This is usually used for character animation. Linear interpolation creates more of a robotic movement.

**Keyframes for other properties**

Godot doesn’t restrict to only edit transform properties. Every property can be used as a track where you can set keyframes.

If you select your sprite while the animation panel is visible, you get a small keyframe button for all of the sprite’s properties. Click on this button and Godot automatically adds a track and keyframe to the current animation.

**Edit keyframes**

For advanced use and to edit keyframe in detail, enable the keyframe editor ( ). This adds an editor pane on the right side of the track settings. When you select a keyframe, you can directly edit its values in this editor:

Additionally, you can also edit the transition value for this keyframe:

This tells Godot, how to change the property values when it reaches this keyframe.

You usually tweak your animations this way, when the movement doesn’t “look right”.

**Advanced: Call Func tracks**

Godot’s animation engine doesn’t stop here. If you’re already comfortable with Godot’s scripting language GDScript and API you know that each node type is a class and has a bunch of callable functions.

For example, the `SamplePlayer2D` node type has a function to play a sample.
Fig. 4.18: Keyframes for other properties

Fig. 4.19: Keyframe editor editing a key

Fig. 4.20: Keyframe editor editing a transition
 Wouldn’t it be great to play a sample at a specific keyframe in an animation? This is where “Call Func Tracks” come in handy. These tracks reference a node again, this time without a reference to a property. Instead a keyframe holds the name and arguments of a function, that Godot should call when reaching this keyframe.

To let Godot play a sample when it reaches a keyframe, follow this list:

Add a SamplePlayer2D to the Scene Tree and add a sample library and a sample to it.

![Add SamplePlayer2D](image)

Fig. 4.21: Add SamplePlayer2D

Click on “Add track” on the animation panel’s track controls.

Select “Add Call Func Track” from the list of possible track types.

![Add Call Func Track](image)

Fig. 4.22: Add Call Func Track

Select the SamplePlayer2D node in the selection window. Godot adds the track with the reference to the node.

Select the timeline position, where Godot should play the sample by clicking on the timeline header.
Enable the Keyframe Editor by clicking on the Keyframe Editor icon.

Click on “Add keyframe” near the settings of our func track and select the keyframe.

Enter “play” as the function name and set the argument counter to 1. Select “String” as the first argument type and use the sample name as the argument value.

When Godot reaches the keyframe, Godot calls the SamplePlayer2D node’s “play” function with the sample name.
Fig. 4.25: Keyframe settings of a call func track

References

Animation panel reference

The animation panel has the following parts (from left to right):

Animation controls

- Play animation backwards from current position
- Play animation backwards from the animation end
- Stop animation
- Play animation forwards from the animation beginning
- Play animation forwards from the current position
- Direct time selection
Animation management

• Create a new animation
• Load animation
• Save animation
• Duplicate animation
• Rename animation
• Delete animation
• Animation selection
• Automatically play selected animation
• Edit animation blend times
• Extended animation Tools

Timeline zoom level control

• Length of animation
4.8.2 Cutout animation

What is it?

Cut-out is a technique of animating in 2D where pieces of paper (or similar material) are cut in special shapes and laid one over the other. The papers are animated and photographed, frame by frame using a stop motion technique (more info here).

With the advent of the digital age, this technique became possible using computers, which resulted in an increased amount of animation TV shows using digital Cut-out. Notable examples are South Park or Jake and the Never Land Pirates.

In video games, this technique also become very popular. Examples of this are Paper Mario or Rayman Origins.
**Cutout in Godot**

Godot provides a few tools for working with these kind of assets, but its overall design makes it ideal for the workflow. The reason is that, unlike other tools meant for this, Godot has the following advantages:

- **The animation system is fully integrated with the engine**: This means, animations can control much more than just motion of objects, such as textures, sprite sizes, pivots, opacity, color modulation, etc. Everything can be animated and blended.

- **Mix with Traditional**: AnimatedSprite allows traditional animation to be mixed, very useful for complex objects, such as shape of hands and foot, changing facial expression, etc.

- **Custom Shaped Elements**: Can be created with *Polygon2D* allowing the mixing of UV animation, deformations, etc.

- **Particle Systems**: Can also be mixed with the traditional animation hierarchy, useful for magic effects, jetpacks, etc.

- **Custom Colliders**: Set colliders and influence areas in different parts of the skeletons, great for bosses, fighting games, etc.

- **Animation Tree**: Allows complex combinations and blendings of several animations, the same way it works in 3D.

And much more!

**Making of GBot!**

For this tutorial, we will use as demo content the pieces of the GBot character, created by Andreas Esau.

Get your assets: *gbot_resources.zip*.

**Setting up the rig**

Create an empty Node2D as root of the scene, we will work under it:

OK, the first node of the model that we will create will be the hip. Generally, both in 2D and 3D, the hip is the root of the skeleton. This makes it easier to animate:
Next will be the torso. The torso needs to be a child of the hip, so create a child sprite and load the torso, later accommodate it properly:

This looks good. Let’s see if our hierarchy works as a skeleton by rotating the torso:

Ouch, that doesn’t look good! The rotation pivot is wrong, this means it needs to be adjusted.

This small little cross in the middle of the _Sprite_ is the rotation pivot:
Adjusting the pivot

The pivot can be adjusted by changing the *offset* property in the Sprite:
However, there is a way to do it more *visually*. While hovering over the desired pivot point, simply press the “v” key to move the pivot there for the selected Sprite. Alternately, there is a tool in the tool bar that has a similar function.

Now it looks good! Let’s continue adding body pieces, starting by the right arm. Make sure to put the sprites in hierarchy, so their rotations and translations are relative to the parent:
This seems easy, so continue with the left arm. The rest should be simple! Or maybe not:

Godot does not support cutout rigs after all. Come back next year, maybe for 3.0.. no wait. Just Kidding! It works just fine.

But how can this problem be solved? We want the left arm to appear behind the hip and the torso. For this, we can move the nodes behind the hip (note that you can bypass this by setting the Node2D Z property, but then you won’t learn about all this!):

But then, we lose the hierarchy layout, which allows to control the skeleton like.. a skeleton. Is there any hope?.. Of Course!

**RemoteTransform2D node**

Godot provides a special node, *RemoteTransform2D*. This node will transform nodes that are sitting somewhere else in the hierarchy, by applying the transform to the remote nodes.

This enables to have a visibility order independent from the hierarchy.

Simply create two more nodes as children from torso, remote_arm_l and remote_hand_l and link them to the actual sprites:
Moving the remote transform nodes will move the sprites, allowing you to easily animate and pose the character:

**Completing the skeleton**

Complete the skeleton by following the same steps for the rest of the parts. The resulting scene should look similar to this:
The resulting rig will be easy to animate. By selecting the nodes and rotating them you can animate forward kinematics (FK) efficiently.

For simple objects and rigs this is fine, however the following problems are common:

- Selecting sprites can become difficult for complex rigs, and the scene tree ends being used due to the difficulty of clicking over the proper sprite.
- Inverse Kinematics is often desired for extremities.

To solve these problems, Godot supports a simple method of skeletons.

**Skeletons**

Godot doesn’t actually support *true* Skeletons, but it does feature a helper to create “bones” between nodes. This is enough for most cases, but the way it works is not completely obvious.

As an example, let’s turn the right arm into a skeleton. To create skeletons, a chain of nodes must be selected from top to bottom:
Then, the option to create a skeleton is located at Edit > Make Bones:

This will add bones covering the arm, but the result is not quite what is expected.
It looks like the bones are shifted up in the hierarchy. The hand connects to the arm, and the arm to the body. So the question is:

- Why does the hand lack a bone?
- Why does the arm connect to the body?

This might seem strange at first, but will make sense later on. In traditional skeleton systems, bones have a position, an orientation and a length. In Godot, bones are mostly helpers so they connect the current node with the parent. Because of this, **toggling a node as a bone will just connect it to the parent.**

So, with this knowledge. Let’s do the same again so we have an actual, useful skeleton.

The first step is creating an endpoint node. Any kind of node will do, but *Position2D* is preferred because it’s visible in the editor. The endpoint node will ensure that the last bone has orientation.
Now select the whole chain, from the endpoint to the arm and create bones:
The result resembles a skeleton a lot more, and now the arm and forearm can be selected and animated. Finally, create endpoints in all meaningful extremities and connect the whole skeleton with bones up to the hip:
Finally! the whole skeleton is rigged! On close look, it is noticeable that there is a second set of endpoints in the hands. This will make sense soon.

Now that a whole skeleton is rigged, the next step is setting up the IK chains. IK chains allow for more natural control of extremities.

**IK chains**

IK chains are a powerful animation tool. Imagine you want to pose a character’s foot in a specific position on the ground. Without IK chains, each motion of the foot would require rotating and positioning several other bones. This would be quite complex and lead to imprecise results.

What if we could move the foot and let the rest of the leg self-adjust?

This type of posing is called IK (Inverse Kinematic).

To create an IK chain, simply select a chain of bones from endpoint to the base for the chain. For example, to create an IK chain for the right leg, select the following:
Then enable this chain for IK. Go to Edit > Make IK Chain.

As a result, the base of the chain will turn *Yellow*.
Once the IK chain is set-up, simply grab any of the bones in the extremity, any child or grand-child of the base of the chain and try to grab it and move it. Result will be pleasant, satisfaction warranted!

**Animation**

The following section will be a collection of tips for creating animation for your rigs. If unsure about how the animation system in Godot works, refresh it by checking again the *Animations*.

**2D animation**

When doing animation in 2D, a helper will be present in the top menu. This helper only appears when the animation editor window is opened:
The key button will insert location/rotation/scale keyframes to the selected objects or bones. This depends on the mask enabled. Green items will insert keys while red ones will not, so modify the key insertion mask to your preference.

**Rest pose**

These kind of rigs do not have a “rest” pose, so it’s recommended to create a reference rest pose in one of the animations.

Simply do the following steps:

1. Make sure the rig is in “rest” (not doing any specific pose).
2. Create a new animation, rename it to “rest”.
3. Select all nodes (box selection should work fine).
4. Select “loc” and “rot” on the top menu.
5. Push the key button. Keys will be inserted for everything, creating a default pose.
Rotation

Animating these models means only modifying the rotation of the nodes. Location and scale are rarely used, with the only exception of moving the entire rig from the hip (which is the root node).

As a result, when inserting keys, only the “rot” button needs to be pressed most of the time:

This will avoid the creation of extra animation tracks for the position that will remain unused.
Keyframing IK

When editing IK chains, it is not necessary to select the whole chain to add keyframes. Selecting the endpoint of the chain and inserting a keyframe will automatically insert keyframes until the chain base too. This makes the task of animating extremities much simpler.

Moving sprites above and behind others.

RemoteTransform2D works in most cases, but sometimes it is really necessary to have a node above and below others during an animation. To aid on this the “Behind Parent” property exists on any Node2D:

Batch setting transition curves

When creating really complex animations and inserting lots of keyframes, editing the individual keyframe curves for each can become an endless task. For this, the Animation Editor has a small menu where changing all the curves is easy. Just select every single keyframe and (generally) apply the “Out-In” transition curve to smooth the animation:
4.9 Lighting

4.9.1 Lighting

Introduction

Lights emit light that mix with the materials and produces a visible result. Light can come from several types of sources in a scene:

- From the Material itself, in the form of the emission color (though it does not affect nearby objects unless baked).
- Light Nodes: Directional, Omni and Spot.
- Ambient Light in the Environment.
- Baked Light (read doc_light_baking).

The emission color is a material property. You can read more about it in the Fixed materials tutorial.

Light nodes

As mentioned before, there are three types of light nodes: Directional, Omni and Spot. Each has different uses and will be described in detail below, but first let’s take a look at the common parameters for lights:
Each one has a specific function:

- **Enabled**: Light is emitted only if this flag is set.
- **Bake Mode**: When using the light baker, the role of this light can be defined in this enumerator. The role will be followed even if the light is disabled, which allows to configure a light and then disable it for baking.
- **Energy**: This value is a multiplier for the light, it’s specially useful for *High dynamic range* and for Spot and Omni lights, because it can create very bright spots near the emitter.
- **Diffuse and Specular**: These light values get multiplied by the material light and diffuse colors. A white value does not mean that light will be white, but that the material color will be kept.
- **Operator**: It is possible to make some lights negative for a darkening effect.
- **Projector**: Lights can project a texture for the diffuse light (currently only supported in Spot light).

**Directional light**

This is the most common type of light and represents a light source very far away (such as the sun). It is also the cheapest light to compute and should be used whenever possible (although it’s not the cheapest shadow-map to compute, but more on that later).

Directional light models an infinite number of parallel light rays covering the whole scene. The directional light node is represented by a big arrow, which indicates the direction of the light rays. However, the position of the node does not affect the lighting at all, and can be anywhere.
Every face whose front-side is hit by the light rays is lit, the others stay dark. Most light types have specific parameters but directional lights are pretty simple in nature so they don’t.

**Omni light**

Omni light is a point source that emits light spherically in all directions up to a given radius (distance from the node’s position). The radius is a parameter of the light and can be controlled by the user. Just as in real life, the intensity of omni light decreases with the distance and vanishes at the defined radius. Omni light sources should be used to represent lamps or bulbs or any other light source that originates approximately in a point.
In reality, the attenuation of omni light is proportional to the squared distance from the point source. This can be easily understood if you imagine a sphere around the omni light with a certain radius. No matter how large the sphere is, the number of rays passing through it is always the same. If the radius of the sphere is doubled, the area of the sphere increases by four. In other words, the density of rays (the number of rays per square area) decreases quadratically with the distance.

Inverse-quadratic attenuation curves are inconvenient for artists: they never reach zero and have almost infinitely large values near the emitter. So Godot simulates omni light with an artist-controlled exponential curve instead.
4.9. Lighting
Spot light

Spot lights are similar to omni lights, except they emit light only into a cone (or “cutoff”). They are useful to simulate flashlights, car lights, etc. This kind of light is also attenuated towards the opposite direction it points to.

Ambient light

Ambient light can be found in the properties of a WorldEnvironment (remember only one of such can be instanced per scene). Ambient light consists of a uniform light and energy. This light is applied the same to every single pixel of the rendered scene, except to objects that used baked light.

Baked light

Baked light stands for pre-computed ambient light. It can serve multiple purposes, such as baking light emitters that are not going to be used in real-time, and baking light bounces from real-time lights to add more realism to a scene (see doc_light_baking tutorial for more information).

4.9.2 Shadow mapping
Introduction

Simply throwing a light is not enough to realistically illuminate a scene. It should be, in theory, but given the way video hardware works, parts of objects that should not be reached by light are lit anyway.

Most people (including artists), see shadows as something projected by light, as if they were created by the light itself by darkening places that are hidden from the light source.

This is actually not correct and it’s important to understand that shadows are places where light simply does not reach. As a rule (and without counting indirect light) if a light is turned off, the places where shadow appear should remain the same. In other words, shadows should not be seen as something “added” to the scene, but as an area that “remains dark”.

All light types in Godot can use shadow mapping, and all support several different techniques that trade quality by performance. Shadow mapping uses a texture storing the “depth view” of the light and checks against it in real-time for each pixel it renders.

The bigger the resolution of the shadow map texture, the more detail the shadow has, but more video memory and bandwidth consumed (which means frame-rate goes down).

Shadows by light type

Directional light shadows

Directional lights can affect a really big area. The bigger the scene, the bigger the affected area. Given the shadow map resolution stays the same, the same amount of shadow pixels cover a bigger area, resulting in blocky shadows. Multiple techniques exist to deal with resolution problems, but the most common one is PSSM (Parallel Split Shadow Maps):
These techniques divide the view in 2 or 4 sections, and a shadow is rendered for each. This way, close objects can use larger shadow while further away objects will use one in less detail, but in proportion this seems to make the shadow map size increase while it’s actually kept the same. Of course, this technique is not free, the more splits the more the performance goes down. On mobile, it is generally inconvenient to use more than 2 splits.

An alternative technique is PSM (Perspective Shadow Mapping). This technique is much cheaper than PSSM (as cheap as orthogonal), but it only really works for a few camera angles respect to the light. In other words, PSM is only useful for games where the camera direction and light direction are both fixed, and the light is not parallel to the camera (which is when PSM completely breaks).

**Omnidirectional lights**

Omnidirectional lights are also troublesome. How to represent 360 degrees of light with a single texture? There are two alternatives, the first one is to use DPSM (Dual Paraboloid Shadow Mapping). This technique is fast, but it requires DISCARD to be used (which makes it not very usable on mobile). DPSM can also look rather bad if the geometry is not tessellated enough, so more vertices might be necessary if it doesn’t look tight. The second option is to simply not use a shadow map, and use a shadow cubemap. This is faster, but requires six passes to render all directions and is not supported on the current (GLES2) renderer.

As few considerations when using DPSM shadow maps:

- Keep Slope-Scale on 0.
- Use a small value for Z-Offset, if this look wrong, make it smaller.
- ESM filtering can improve the look.
- The seams between the two halves of the shadow are generally noticeable, so rotate the light to make them show less.
Spot light shadows

Spot light shadows are generally the simpler, just needing a single texture and no special techniques.

Shadows parameters

The fact that shadows are actually a texture can generate several problems. The most common is Z fighting (lines at the edge of the objects that cast the shadows. There are two ways to fix this, the first is to tweak the offset parameters, and the second is to use a filtered shadow algorithm, which generally looks better and has not as many glitches, but consumes more GPU time.

Adjusting z-offset

So, you have decided to go with non-filtered shadows because they are faster, you want a little more detail or maybe you just like the sexy saw-like shadow outlines because they remind you of your favorite previous-gen games. Truth is, this can be kind of be a pain, but most of the time it can be adjusted to have nice results. There is no magic number and whatever result you come up will be different from scene to scene, it just takes a while of tweaking. Let’s go step by step.

First step is to turn on the shadows, let’s assume that both Z-Offset and Z-Slope-Scale are at 0. You will be greeted by this:
Holy crap, the shadow is all over the place and extremely glitchy! This happens because the shadow is fighting with the same geometry that is casting it. This is called “self-shadowing”. To avoid this meaningless fight, you realize you need to make peace between the shadow and the geometry, so you push back the shadow a little by increasing the shadow Z-Offset. This improves things a lot:

But it’s not quite perfect, self shadowing did not disappear completely. So close to perfection but still not there.. so in a turn of greed you increase the Z-Offset even more!
And it gets rid of those self-shadowings! Hooray! Except something is wrong.. oh, right. Being pushed back too much, the shadows start disconnecting from their casters, which looks pretty awful. Ok, you go back to the previous Z-offset.

This is when Z-Slope-Scale comes to save the day. This setting makes shadow caster objects thinner, so the borders don’t self-shadow:
Aha! Finally something that looks acceptable. It’s perfectly acceptable and you can perfectly ship a game that looks like this (imagine you are looking at Final Fantasy quality art btw, not this horrible attempt at 3D modelling). There may be very tiny bits left of self shadowing that no one cares about, so your inextinguishable greed kicks in again and you raise the Z-Slope Scale again:
Well, that was too much, shadows casted are way too thin and don’t look good anymore. Well, though luck, the previous setting was good anyway, let’s accept that perfection does not exist and move on to something else.

**Important!**

If you are using shadow maps with directional lights, make sure that the view distance of the camera is set to an optimal range. This means, if the distance between your camera and the visible end of the scene is 100, then set the view distance to that value. If a greater than necessary value is used, the shadow maps will lose detail as they will try to cover a bigger area.

So, always make sure to use the optimal range!

**Shadow filtering**

Raw shadows are blocky. Increasing their resolution just makes smaller blocks, but they are still blocks.

Godot offers a few ways to filter them (shadow in the example is low-resolution on purpose!):
4.9. Lighting

no filtering

pcf5

pcf13
PCF5 and PCF13 are simple texture-space filtering. Will make the texture a little more acceptable but still needs considerable resolution for it to look good.

ESM is a more complex filter and has a few more tweaking parameters. ESM uses shadow blurring (amount of blur passes and multiplier can be adjusted).

### 4.10 Shading

#### 4.10.1 Shading language

**Introduction**

Godot uses a simplified shader language (almost a subset of GLSL). Shaders can be used for:

- Materials
- Post-Processing
- 2D

and are divided in *Vertex, Fragment and Light sections.*

**Language**

**Typing**

The language is statically type and supports only a few operations. Arrays, classes, structures, etc are not supported. Several built-in datatypes are provided:

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>Void</td>
</tr>
<tr>
<td>bool</td>
<td>boolean (true or false)</td>
</tr>
<tr>
<td>float</td>
<td>floating point</td>
</tr>
<tr>
<td>vec2</td>
<td>2-component vector, float subindices (x,y or r,g)</td>
</tr>
<tr>
<td>vec3</td>
<td>3-component vector, float subindices (x,y,z or r,g,b)</td>
</tr>
<tr>
<td>vec4, color</td>
<td>4-component vector, float subindices (x,y,z,w or r,g,b,a)</td>
</tr>
<tr>
<td>mat2</td>
<td>2x2 matrix, vec3 subindices (x,y)</td>
</tr>
<tr>
<td>mat3</td>
<td>3x3 matrix, vec3 subindices (x,y,z)</td>
</tr>
<tr>
<td>mat4</td>
<td>4x4 matrix, vec4 subindices (x,y,z,w)</td>
</tr>
<tr>
<td>texture</td>
<td>texture sampler, can only be used as uniform</td>
</tr>
<tr>
<td>cubemap</td>
<td>cubemap sampler, can only be used as uniform</td>
</tr>
</tbody>
</table>

**Syntax**

The syntax is similar to C, with statements ending with ; and comments as // and /* */. Example:

```plaintext
float a = 3;
vec3 b;
b.x = a;
```
Swizzling

It is possible to use swizzling to reassigning subindices or groups of subindices, in order:

```
vec3 a = vec3(1,2,3);
vec3 b = a.zyx; // b will contain vec3(3,2,1)
vec2 c = a.xy; // c will contain vec2(1,2)
vec4 d = a.xyzz; // d will contain vec4(1,2,3,3)
```

Constructors

Constructors take the regular amount of elements, but can also accept less if the element has more subindices, for example:

```
vec3 a = vec3(1,vec2(2,3));
vec3 b = vec3(a);
vec3 c = vec3(vec2(2,3),1);
vec4 d = vec4(a,5);
mat3 m = mat3(a,b,c);
```

Conditionals

For now, only the `if` conditional is supported. Example:

```
if (a < b) {
    c = b;
}
```

Uniforms

A variable can be declared as uniform. In this case, its value will come from outside the shader (it will be the responsibility of the material or whatever using the shader to provide it).

```
uniform vec3 direction;
uniform color tint;

vec3 result = tint.rgb * direction;
```

Functions

Simple support for functions is provided. Functions can’t access uniforms or other shader variables.

```
vec3 addtwo(vec3 a, vec3 b) {
   return a+b;
}

vec3 c = addtwo(vec3(1,1,1), vec3(2,2,2));
```
## Built-in functions

Several built-in functions are provided for convenience, listed as follows:

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>float sin ( float )</td>
<td>Sine</td>
</tr>
<tr>
<td>float cos ( float )</td>
<td>Cosine</td>
</tr>
<tr>
<td>float tan ( float )</td>
<td>Tangent</td>
</tr>
<tr>
<td>float asin ( float )</td>
<td>arc-Sine</td>
</tr>
<tr>
<td>float acos ( float )</td>
<td>arc-Cosine</td>
</tr>
<tr>
<td>float atan ( float )</td>
<td>arc-Tangent</td>
</tr>
<tr>
<td>vec_type pow ( vec_type, float )</td>
<td>Power</td>
</tr>
<tr>
<td>vec_type pow ( vec_type, vec_type )</td>
<td>Power (Vec. Exponent)</td>
</tr>
<tr>
<td>vec_type exp ( vec_type )</td>
<td>Base-e Exponential</td>
</tr>
<tr>
<td>vec_type log ( vec_type )</td>
<td>Natural Logarithm</td>
</tr>
<tr>
<td>vec_type sqrt ( vec_type )</td>
<td>Square Root</td>
</tr>
<tr>
<td>vec_type abs ( vec_type )</td>
<td>Absolute</td>
</tr>
<tr>
<td>vec_type sign ( vec_type )</td>
<td>Sign</td>
</tr>
<tr>
<td>vec_type floor ( vec_type )</td>
<td>Floor</td>
</tr>
<tr>
<td>vec_type trunc ( vec_type )</td>
<td>Trunc</td>
</tr>
<tr>
<td>vec_type ceil ( vec_type )</td>
<td>Ceiling</td>
</tr>
<tr>
<td>vec_type fract ( vec_type )</td>
<td>Fractional</td>
</tr>
<tr>
<td>vec_type mod ( vec_type,vec_type )</td>
<td>Remainder</td>
</tr>
<tr>
<td>vec_type min ( vec_type,vec_type )</td>
<td>Minimum</td>
</tr>
<tr>
<td>vec_type max ( vec_type,vec_type )</td>
<td>Maximum</td>
</tr>
<tr>
<td>vec_type clamp ( vec_type value,vec_type min, vec_type max )</td>
<td>Clamp to Min-Max</td>
</tr>
<tr>
<td>vec_type mix ( vec_type a,vec_type b, float c )</td>
<td>Linear Interpolate</td>
</tr>
<tr>
<td>vec_type mix ( vec_type a,vec_type b, vec_type c )</td>
<td>Linear Interpolate (Vector Coef.)</td>
</tr>
<tr>
<td>vec_type step ( vec_type a,vec_type b)</td>
<td>‘ a[i] &lt; b[i] ? 0.0 : 1.0’</td>
</tr>
<tr>
<td>vec_type smoothstep ( vec_type a,vec_type b,vec_type c)</td>
<td></td>
</tr>
<tr>
<td>float length ( vec_type )</td>
<td>Vector Length</td>
</tr>
<tr>
<td>float distance ( vec_type, vec_type )</td>
<td>Distance between vector.</td>
</tr>
<tr>
<td>float dot ( vec_type, vec_type )</td>
<td>Dot Product</td>
</tr>
<tr>
<td>vec3 cross ( vec3, vec3 )</td>
<td>Cross Product</td>
</tr>
<tr>
<td>vec_type normalize ( vec_type )</td>
<td>Normalize to unit length</td>
</tr>
<tr>
<td>vec3 reflect ( vec3, vec3 )</td>
<td>Reflect</td>
</tr>
<tr>
<td>color tex ( texture, vec2 )</td>
<td>Read from a texture in normalized coords</td>
</tr>
<tr>
<td>color texcube ( texture, vec3 )</td>
<td>Read from a cubemap</td>
</tr>
<tr>
<td>vec3 texscreen ( vec2 )</td>
<td>Read RGB from screen (generates a copy)</td>
</tr>
</tbody>
</table>

## Built-in variables

Depending on the shader type, several built-in variables are available, listed as follows:
Material (3D) - VertexShader

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>const vec3 SRC_VERTEX</td>
<td>Model-Space Vertex</td>
</tr>
<tr>
<td>const vec3 SRC_NORMAL</td>
<td>Model-Space Normal</td>
</tr>
<tr>
<td>const vec3 SRC_TANGENT</td>
<td>Model-Space Tangent</td>
</tr>
<tr>
<td>const float SRC_BINORMAL</td>
<td>Direction to Compute Binormal</td>
</tr>
<tr>
<td>vec3 VERTEX</td>
<td>View-Space Vertex</td>
</tr>
<tr>
<td>vec3 NORMAL</td>
<td>View-Space Normal</td>
</tr>
<tr>
<td>vec3 TANGENT</td>
<td>View-Space Tangent</td>
</tr>
<tr>
<td>vec3 BINORMAL</td>
<td>View-Space Binormal</td>
</tr>
<tr>
<td>vec2 UV</td>
<td>UV</td>
</tr>
<tr>
<td>vec2 UV2</td>
<td>UV2</td>
</tr>
<tr>
<td>color COLOR</td>
<td>Vertex Color</td>
</tr>
<tr>
<td>out vec4 VAR1</td>
<td>Varying 1 Output</td>
</tr>
<tr>
<td>out vec4 VAR2</td>
<td>Varying 2 Output</td>
</tr>
<tr>
<td>out float SPEC_EXP</td>
<td>Specular Exponent (for Vertex Lighting)</td>
</tr>
<tr>
<td>out float POINT_SIZE</td>
<td>Point Size (for points)</td>
</tr>
<tr>
<td>const mat4 WORLD_MATRIX</td>
<td>Object World Matrix</td>
</tr>
<tr>
<td>const mat4 INV_CAMERA_MATRIX</td>
<td>Inverse Camera Matrix</td>
</tr>
<tr>
<td>const mat4 PROJECTION_MATRIX</td>
<td>Projection Matrix</td>
</tr>
<tr>
<td>const mat4 MODELVIEW_MATRIX</td>
<td>(InvCamera * Projection)</td>
</tr>
<tr>
<td>const float INSTANCE_ID</td>
<td>Instance ID (for multimesh)</td>
</tr>
<tr>
<td>const float TIME</td>
<td>Time (in seconds)</td>
</tr>
</tbody>
</table>
### Material (3D) - FragmentShader

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>const vec3 VERTEX</td>
<td>View-Space vertex</td>
</tr>
<tr>
<td>const vec4 POSITION</td>
<td>View-Space Position</td>
</tr>
<tr>
<td>const vec3 NORMAL</td>
<td>View-Space Normal</td>
</tr>
<tr>
<td>const vec3 TANGENT</td>
<td>View-Space Tangent</td>
</tr>
<tr>
<td>const vec3 BINORMAL</td>
<td>View-Space Binormal</td>
</tr>
<tr>
<td>const vec3 NORMALMAP</td>
<td>Alternative to NORMAL, use for normal texture output.</td>
</tr>
<tr>
<td>const vec3 NORMALMAP_DEPTH</td>
<td>Complementary to the above, allows changing depth of normalmap.</td>
</tr>
<tr>
<td>const vec2 UV</td>
<td>UV</td>
</tr>
<tr>
<td>const vec2 UV2</td>
<td>UV2</td>
</tr>
<tr>
<td>const color COLOR</td>
<td>Vertex Color</td>
</tr>
<tr>
<td>const vec4 VARI</td>
<td>Varying 1</td>
</tr>
<tr>
<td>const vec4 VAR2</td>
<td>Varying 2</td>
</tr>
<tr>
<td>const vec2 SCREEN_UV</td>
<td>Screen Texture Coordinate (for using with texscreen)</td>
</tr>
<tr>
<td>const float TIME</td>
<td>Time (in seconds)</td>
</tr>
<tr>
<td>const vec2 POINT_COORD</td>
<td>UV for point, when drawing point sprites.</td>
</tr>
<tr>
<td>out vec3 DIFFUSE</td>
<td>Diffuse Color</td>
</tr>
<tr>
<td>out vec4 DIFFUSE_ALPHA</td>
<td>Diffuse Color with Alpha (using this sends geometry to alpha pipeline)</td>
</tr>
<tr>
<td>out vec3 SPECULAR</td>
<td>Specular Color</td>
</tr>
<tr>
<td>out vec3 EMISSION</td>
<td>Emission Color</td>
</tr>
<tr>
<td>out float SPEC_EXP</td>
<td>Specular Exponent (Fragment Version)</td>
</tr>
<tr>
<td>out float GLOW</td>
<td>Glow</td>
</tr>
<tr>
<td>out mat4 INV_CAMERA_MATRIX</td>
<td>Inverse camera matrix, can be used to obtain world coords (see example below).</td>
</tr>
</tbody>
</table>

### Material (3D) - LightShader

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>const vec3 NORMAL</td>
<td>View-Space normal</td>
</tr>
<tr>
<td>const vec3 LIGHT_DIR</td>
<td>View-Space Light Direction</td>
</tr>
<tr>
<td>const vec3 EYE_VEC</td>
<td>View-Space Eye-Point Vector</td>
</tr>
<tr>
<td>const vec3 DIFFUSE</td>
<td>Material Diffuse Color</td>
</tr>
<tr>
<td>const vec3 LIGHT_DIFFUSE</td>
<td>Light Diffuse Color</td>
</tr>
<tr>
<td>const vec3 SPECULAR</td>
<td>Material Specular Color</td>
</tr>
<tr>
<td>const vec3 LIGHT_SPECULAR</td>
<td>Light Specular Color</td>
</tr>
<tr>
<td>const float SPECULAR.EXP</td>
<td>Specular Exponent</td>
</tr>
<tr>
<td>const vec1 SHADE_PARAM</td>
<td>Generic Shade Parameter</td>
</tr>
<tr>
<td>const vec2 POINT_COORD</td>
<td>Current UV for Point Sprite</td>
</tr>
<tr>
<td>out vec2 LIGHT</td>
<td>Resulting Light</td>
</tr>
<tr>
<td>const float TIME</td>
<td>Time (in seconds)</td>
</tr>
</tbody>
</table>
### CanvasItem (2D) - VertexShader

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>const vec2 SRC_VERTEX</td>
<td>CanvasItem space vertex.</td>
</tr>
<tr>
<td>vec2 UV</td>
<td>UV</td>
</tr>
<tr>
<td>out vec2 VERTEX</td>
<td>Output LocalSpace vertex.</td>
</tr>
<tr>
<td>out vec2 WORLD_VERTEX</td>
<td>Output WorldSpace vertex.  (use this or the one above)</td>
</tr>
<tr>
<td>color COLOR</td>
<td>Vertex Color</td>
</tr>
<tr>
<td>out vec4 VAR1</td>
<td>Varying 1 Output</td>
</tr>
<tr>
<td>out vec4 VAR2</td>
<td>Varying 2 Output</td>
</tr>
<tr>
<td>out float POINT_SIZE</td>
<td>Point Size (for points)</td>
</tr>
<tr>
<td>const mat4 WORLD_MATRIX</td>
<td>Object World Matrix</td>
</tr>
<tr>
<td>const mat4 EXTRA_MATRIX</td>
<td>Extra (user supplied) matrix via CanvasItem.draw_set_transform(). Identity by default.</td>
</tr>
<tr>
<td>const mat4 PROJECTION_MATRIX</td>
<td>Projection Matrix (model coords to screen).</td>
</tr>
<tr>
<td>const float TIME</td>
<td>Time (in seconds)</td>
</tr>
<tr>
<td>const bool AT_LIGHT_PASS</td>
<td>Whether the shader is being run for a lighting pass (happens per affecting light)</td>
</tr>
</tbody>
</table>

### CanvasItem (2D) - FragmentShader

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>const vec4 SRC_COLOR</td>
<td>Vertex color</td>
</tr>
<tr>
<td>const vec4 POSITION</td>
<td>Screen Position</td>
</tr>
<tr>
<td>vec2 UV</td>
<td>UV</td>
</tr>
<tr>
<td>out color COLOR</td>
<td>Output Color</td>
</tr>
<tr>
<td>out vec3 NORMAL</td>
<td>Optional Normal (used for 2D Lighting)</td>
</tr>
<tr>
<td>out vec3 NORMALMAP</td>
<td>Optional Normal in standard normalmap format (flipped y and Z from 0 to 1)</td>
</tr>
<tr>
<td>out float NORMALMAP_DEPTH</td>
<td>Depth option for above normalmap output, default value is 1.0</td>
</tr>
<tr>
<td>const texture TEXTURE</td>
<td>Current texture in use for CanvasItem</td>
</tr>
<tr>
<td>const vec2 TEXTURE_PIXEL_SIZE</td>
<td>Pixel size for current 2D texture</td>
</tr>
<tr>
<td>in vec4 VAR1</td>
<td>Varying 1 Output</td>
</tr>
<tr>
<td>in vec4 VAR2</td>
<td>Varying 2 Output</td>
</tr>
<tr>
<td>const vec2 SCREEN_UV</td>
<td>Screen Texture Coordinate (for using with texscreen)</td>
</tr>
<tr>
<td>const vec2 POINT_COORD</td>
<td>Current UV for Point Sprite</td>
</tr>
<tr>
<td>const float TIME</td>
<td>Time (in seconds)</td>
</tr>
<tr>
<td>const bool AT_LIGHT_PASS</td>
<td>Whether the shader is being run for a lighting pass (happens per affecting light)</td>
</tr>
</tbody>
</table>
CanvasItem (2D) - LightShader

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>const vec4 POSITION</td>
<td>Screen Position</td>
</tr>
<tr>
<td>in vec3 NORMAL</td>
<td>Input Normal</td>
</tr>
<tr>
<td>in vec2 UV</td>
<td>UV</td>
</tr>
<tr>
<td>in color COLOR</td>
<td>Input Color</td>
</tr>
<tr>
<td>const texture TEXTURE</td>
<td>Current texture in use for CanvasItem</td>
</tr>
<tr>
<td>const vec2 TEXTURE_PIXEL_SIZE</td>
<td>Pixel size for current 2D texture</td>
</tr>
<tr>
<td>in vec4 VAR1</td>
<td>Varying 1 Output</td>
</tr>
<tr>
<td>in vec4 VAR2</td>
<td>Varying 2 Output</td>
</tr>
<tr>
<td>const vec2 SCREEN_UV</td>
<td>Screen Texture Coordinate (for using with texscreen)</td>
</tr>
<tr>
<td>const vec2 POINT_COORD</td>
<td>Current UV for Point Sprite</td>
</tr>
<tr>
<td>const float TIME</td>
<td>Time (in seconds)</td>
</tr>
<tr>
<td>vec2 LIGHT_VEC</td>
<td>Vector from light to fragment, can be modified to alter shadow computation.</td>
</tr>
<tr>
<td>const float LIGHT_HEIGHT</td>
<td>Height of Light</td>
</tr>
<tr>
<td>const color LIGHT_COLOR</td>
<td>Color of Light</td>
</tr>
<tr>
<td>const color LIGHT_SHADOW_COLOR</td>
<td>Color of Light shadow</td>
</tr>
<tr>
<td>vec2 LIGHT_UV</td>
<td>UV for light image</td>
</tr>
<tr>
<td>color SHADOW</td>
<td>Light shadow color override</td>
</tr>
<tr>
<td>out vec4 LIGHT</td>
<td>Light Output (shader is ignored if this is not used)</td>
</tr>
</tbody>
</table>

Examples

Material that reads a texture, a color and multiples them, fragment program:

```glsl
uniform color modulate;
uniform texture source;

DIFFUSE = modulate.rgb * tex(source, UV).rgb;
```

Material that glows from red to white:

```glsl
DIFFUSE = vec3(1,0,0) + vec3(1,1,1) * mod(TIME, 1.0);
```

Standard Blinn Lighting Shader

```glsl
float NdotL = max(0.0, dot(NORMAL, LIGHT_DIR));
vec3 half_vec = normalize(LIGHT_DIR + EYE_VEC);
float eye_light = max(dot(NORMAL, half_vec), 0.0);
LIGHT = LIGHT_DIFFUSE + DIFFUSE + NdotL;
if (NdotL > 0.0) {
    LIGHT += LIGHT_SPECULAR + SPECULAR + pow(eye_light, SPECULAR_EXP);
};
```

Obtaining world-space normal and position in material fragment program:

```glsl
// Use reverse multiply because INV_CAMERA_MATRIX is world2cam
vec4 invcamx = INV_CAMERA_MATRIX.x;
vec4 invcamy = INV_CAMERA_MATRIX.y;
vec4 invcamz = INV_CAMERA_MATRIX.z;
```
vec4 invcamw = INV_CAMERA_MATRIX.w;
mat3 invcam = mat3(invcamx.xyz, invcamy.xyz, invcamz.xyz);
vec3 world_normal = NORMAL * invcam;
vec3 world_pos = (VERTEX - invcamw.xyz) * invcam;

Notes

• Do not use DIFFUSE_ALPHA unless you really intend to use transparency. Transparent materials must be sorted by depth and slow down the rendering pipeline. For opaque materials, just use DIFFUSE.
• Do not use DISCARD unless you really need it. Discard makes rendering slower, specially on mobile devices.
• TIME may reset after a while (may last an hour or so), it’s meant for effects that vary over time.
• In general, every built-in variable not used results in less shader code generated, so writing a single giant shader with a lot of code and optional scenarios is often not a good idea.

4.10.2 Shader materials

Introduction

For the most common cases, Fixed materials are enough to create the desired textures or look and feel. Shader materials are a step beyond that, adding a huge amount of flexibility. With them, it is possible to:

• Create procedural textures.
• Create complex texture blendings.
• Create animated materials, or materials that change with time.
• Create refractive effects or other advanced effects.
• Create special lighting shaders for more exotic materials.
• Animate vertices, like tree leaves or grass.
• And much more!

Traditionally, most engines will ask you to learn GLSL, HLSL or CG, which are pretty complex for the skillset of most artists. Godot uses a simplified version of a shader language that will detect errors as you type, so you can see your edited shaders in real-time. Additionally, it is possible to edit shaders using a visual, node-based graph editor.

Creating a ShaderMaterial

Create a new ShaderMaterial in some object of your choice. Go to the “Shader” property, then create a new “MaterialShader” (use “MaterialShaderGraph” for access to the visual graph editor):
Edit the newly created shader, and the shader editor will open:

![Shader Material Editor](image)

There are three code tabs open, the first is for the vertex shader, the second for the fragment and the third for the lighting. The shader language is documented in *Shading language* so a small example will be presented next.

Create a very simple fragment shader that writes a color:

```cpp
uniform color col;
DIFFUSE = col.rgb;
```

Code changes take place in real-time. If the code is modified, it will be instantly recompiled and the object will be updated. If a typo is made, the editor will notify of the compilation failure:
Finally, go back and edit the material, and the exported uniform will be instantly visible:

This allows to very quickly create custom, complex materials for every type of object.

### 4.10.3 Screen-reading shaders

**Introduction**

Very often it is desired to make a shader that reads from the same screen it’s writing to. 3D APIs such as OpenGL or DirectX make this very difficult because of internal hardware limitations. GPUs are extremely parallel, so reading and writing causes all sort of cache and coherency problems. As a result, not even the most modern hardware supports this properly.

The workaround is to make a copy of the screen, or a part of the screen, to a back-buffer and then read from it while drawing. Godot provides a few tools that makes this process easy!
TexScreen shader instruction

Godot Shading language has a special instruction, “texscreen”, it takes as parameter the UV of the screen and returns a vec3 RGB with the color. A special built-in varying: SCREEN_UV can be used to obtain the UV for the current fragment. As a result, this simple 2D fragment shader:

```
COLOR = vec4( texscreen(SCREEn_UV), 1.0 );
```

results in an invisible object, because it just shows what lies behind. The same shader using the visual editor looks like this:

TexScreen example

Texscreen instruction can be used for a lot of things. There is a special demo for Screen Space Shaders, that you can download to see and learn. One example is a simple shader to adjust brightness, contrast and saturation:

```
uniform float brightness = 1.0;
uniform float contrast = 1.0;
uniform float saturation = 1.0;

vec3 c = texscreen(SCREEN_UV);

c.rgb = mix(vec3(0.0), c.rgb, brightness);
c.rgb = mix(vec3(0.5), c.rgb, contrast);
c.rgb = mix(vec3(dot(vec3(1.0), c.rgb)*0.33333), c.rgb, saturation);

COLOR.rgb = c;
```

Behind the scenes

While this seems magical, it’s not. The Texscreen instruction, when first found in a node that is about to be drawn, does a full-screen copy to a back-buffer. Subsequent nodes that use texscreen() in shaders will not have the screen copied for them, because this ends up being very inefficient.

As a result, if shaders that use texscreen() overlap, the second one will not use the result of the first one, resulting in unexpected visuals:
In the above image, the second sphere (top right) is using the same source for `texscreen()` as the first one below, so the first one “disappears”, or is not visible.

To correct this, a `BackBufferCopy` node can be instanced between both spheres. `BackBufferCopy` can work by either specifying a screen region or the whole screen:

With correct back-buffer copying, the two spheres blend correctly:

**Back-buffer logic**

So, to make it clearer, here’s how the backbuffer copying logic works in Godot:

- If a node uses the `texscreen()` method, the entire screen is copied to the back buffer before drawing that node. This only happens the first time, subsequent nodes do not trigger this.

- If a `BackBufferCopy` node was processed before the situation in the point above (even if `texscreen()` was not used), this behavior described in the point above does not happen. In other words, automatic copying of the entire screen only happens if `texscreen()` is used in a node for the first time and no `BackBufferCopy` node (not disabled) was found before in tree-order.
• BackBufferCopy can copy either the entire screen or a region. If set to only a region (not the whole screen) and your shader uses pixels not in the region copied, the result of that read is undefined (most likely garbage from previous frames). In other words, it’s possible to use BackBufferCopy to copy back a region of the screen and then use texscreen() on a different region. Avoid this behavior!

4.11 Networking

4.11.1 HTTP client class

Here’s an example of using the HTTPClient class. It’s just a script, so it can be run by executing:

c:\godot> godot -s http_test.gd

It will connect and fetch a website.

```godot
extends SceneTree

# HTTPClient demo
# This simple class can do HTTP requests, it will not block but it needs to be polled

func _init():
    var err=0
    var http = HTTPClient.new() # Create the Client

    var err = http.connect("www.php.net",80) # Connect to host/port
    assert(err==OK) # Make sure connection was OK

    # Wait until resolved and connected
    while (http.get_status()==HTTPClient.STATUS_CONNECTING or http.get...
        http.poll()
        print("Connecting..")
        OS.delay_msec(500)

    assert( http.get_status() == HTTPClient.STATUS_CONNECTED ) # Could not connect

    # Some headers

    var headers=[
        "User-Agent: Pirulo/1.0 (Godot)",
        "Accept: */*
    ]

    err = http.request(HTTPClient.METHOD_GET,"/ChangeLog-5.php",headers) # Request a...
        assert( err == OK ) # Make sure all is OK

    while (http.get_status() == HTTPClient.STATUS_REQUESTING):
        # Keep polling until the request is going on
        http.poll()
        print("Requesting..")
        OS.delay_msec(500)
```

280 Chapter 4. Engine features


```python
assert http.get_status() == HTTPClient.STATUS_BODY or http.get_status() == HTTPClient.STATUS_CONNECTED # Make sure request finished well.

print("response? ", http.has_response()) # Site might not have a response.

if http.has_response():
    # If there is a response..
    var headers = http.get_response_headers_as_dictionary() # Get response headers
    print("code: ", http.get_response_code()) # Show response code
    print("**headers:
    \n", headers) # Show headers

    # Getting the HTTP Body

    if http.is_response_chunked():
        # Does it use chunks?
        print("Response is Chunked!")
    else:
        # Or just plain Content-Length
        var bl = http.get_response_body_length()
        print("Response Length: ", bl)

        # This method works for both anyway

        var rb = RawArray() # Array that will hold the data

        while http.get_status()==HTTPClient.STATUS_BODY:
            # While there is body left to be read
            http.poll()
            var chunk = http.read_response_body_chunk() # Get a chunk
            if chunk.size()==0:
                # Got nothing, wait for buffers to fill a bit
                OS.delay_usec(1000)
            else:
                rb = rb + chunk # Append to read buffer

        # Done!

        print("bytes got: ", rb.size())
        var text = rb.get_string_from_ascii()
        print("Text: ", text)

quit()
```

4.11.2 SSL certificates

Introduction

It is often desired to use SSL connections for communications to avoid “man in the middle” attacks. Godot has a connection wrapper, `StreamPeerSSL`, which can take a regular connection and add security around it. The `HTTPClient` class also supports HTTPS by using this same wrapper.

For SSL to work, certificates need to be provided. A .crt file must be specified in the project settings:
This file should contain any number of public certificates in http://en.wikipedia.org/wiki/Privacy-enhanced_Electronic_Mail format.

Of course, remember to add .crt as filter so the exporter recognizes this when exporting your project.
There are two ways to obtain certificates:

**Approach 1: self signed cert**

The first approach is the simplest, just generate a private and public key pair, and put the public pair in the .crt file (again, in PEM format). The private key should go to your server.

OpenSSL has some documentation about this. This approach also does not require domain validation nor requires you to spend a considerable amount of money in purchasing certificates from a CA.

**Approach 2: CA cert**

The second approach consists of using a certificate authority (CA) such as Verisign, Geotrust, etc. This is a more cumbersome process, but it’s more “official” and ensures your identity is clearly represented.

Unless you are working with large companies or corporations, or need to connect to someone else’s servers (i.e., connecting to Google or some other REST API provider via HTTPS) this method is not as useful.

Also, when using a CA issued cert, you must enable domain validation, to ensure the domain you are connecting to is the one intended, otherwise any website can issue any certificate in the same CA and it will work.

If you are using Linux, you can use the supplied certs file, generally located in:
/etc/ssl/certs/ca-certificates.crt

This file allows HTTPS connections to virtually any website (i.e., Google, Microsoft, etc.).
Or just pick any of the more specific certificates there if you are connecting to a specific one.

## 4.12 Platform-specific

### 4.12.1 Android in-app purchases

Godot engine has integrated GooglePaymentsV3 module with which we can implement in-app purchases in our game. The Godot engine demo project repository has an android-iap example project. It includes a gdscript interface for android IAP.

Check the repository here [https://github.com/godotengine/godot-demo-projects](https://github.com/godotengine/godot-demo-projects)

Find the iap.gd script in

```
godot-demo-projects/misc/android_iap
```

Add it to the Autoload list and name it as IAP so that we can reference it anywhere in the game.

### Getting the product details

When starting our game, we will need to get the item details from Google such as the product price, description and localized price string etc.

```
#First listen to the sku details update callback
IAP.connect("sku_details_complete",self,"sku_details_complete")

#Then ask google the details for these items
IAP.sku_details_query(["pid1","pid2"])
#pid1 and pid2 are our product ids entered in Googleplay dashboard

#This will be called when sku details are retrieved successfully
func sku_details_complete():
    print(IAP.sku_details) #This will print the details as JSON format, refer the format in iap.gd
    print(IAP.sku_details["pid1"].price) #print formatted localized price
```

We can use the IAP details to display the title, price and/or description on our shop scene.

### Check if user purchased an item

When starting our game, we can check if the user has purchased any product. YOU SHOULD DO THIS ONLY AFTER 2/3 SECONDS AFTER YOUR GAME IS LOADED. If we do this as the first thing when the game is launched, IAP might not be initialized and our game will crash on start.

```
#Add a listener first
IAP.connect("has_purchased",self,"iap_has_purchased")
IAP.request_purchased() #Ask Google for all purchased items
```
# This will call for each and every user purchased products
func iap_has_purchased(item_name):
    print(item_name) #print the name of purchased items

Google IAP policy says the game should restore the user’s purchases if the user replaces his phone or reinstall the same app. We can use the above code to do check what products the user has purchased and we can make our game respond accordingly.

**Simple Purchase**

We can put this purchase logic on a product’s buy button.

```python
# First listen for purchase_success callback
IAP.connect("purchase_success", self, "purchase_success_callback")

# Then call purchase like this
IAP.purchase("pid1") #replace pid1 with your product id
IAP.purchase("pid2") #replace pid2 with your another product id

# This function will be called when the purchase is a success
func purchase_success_callback(item):
    print(item + " has purchased")
```

We can also implement other signals for the purchase flow and improve the user experience as you needed.

- **purchase_fail**: When the purchase is failed due to any reason
- **purchase_cancel**: When the user cancels the purchase
- **purchase_owned**: When the user already bought the product earlier

**Consumables and Non-Consumables**

There are two types of products - consumables and non-consumables. **Consumables** are purchased and used, for eg: healing potions which can be purchased again and again. **Non-consumables** are one time purchases, for eg: Level packs.

Google doesn’t have this separation in their dashboard. If our product is a consumable, and if a user has purchased it, it will not be available for purchase until it is consumed. So we should call the consume method for our consumables and don’t call consume for your non-consumables.

```python
IAP.connect("consume_success", self, "on_consume_success")
IAP.consume("pid")

func on_consume_success(item):
    print(item + " consumed")
```

If our game has only consumables, we don’t have to do this. We can set it to consume the item automatically after a purchase.

```python
IAP.set_auto_consume(true)
```

If our game has only non-consumables, we can

```python
IAP.set_auto_consume(false)
```

We should set the auto consume value only once when the game starts.
Testing

If we add a gmail id as a tester in Google dashboard, that tester can purchase items and he will not be charged. Another way to test IAP is using redeem codes generated by us for our game because the purchase flow is the same.

Third way of testing is in development side. If we put the product ids as shown below, we will get a static fixed response according to the product id. This is a quick way of testing things before going to the dashboard.

- android.test.purchased
- android.test.canceled
- android.test.refunded
- android.test.item_unavailable

4.12.2 Services for iOS

At the moment, there are 2 iOS APIs partially implemented, GameCenter and Storekit. Both use the same model of asynchronous calls explained below.

Asynchronous methods

When requesting an asynchronous operation, the method will look like this:

```python
Error purchase(Variant p_params);
```

The parameter will usually be a Dictionary, with the information necessary to make the request, and the call will have 2 phases. First, the method will immediately return an Error value. If the Error is not ‘OK’, the call operation is completed, with an error probably caused locally (no internet connection, API incorrectly configured, etc). If the error value is ‘OK’, a response event will be produced and added to the ‘pending events’ queue. Example:

```python
func on_purchase_pressed()
    var result = InAppStore.purchase( { "product_id": "my_product" } )
    if result == OK:
        animation.play("busy") # show the "waiting for response" animation
    else:
        show_error()

    # put this on a 1 second timer or something
func check_events()
    while InAppStore.getPendingEventCount() > 0:
        var event = InAppStore.popPendingEvent()
        if event.type == "purchase":
            if event.result == "ok":
                show_success(event.product_id)
            else:
                show_error()
```

Remember that when a call returns OK, the API will always produce an event through the pending_event interface, even if it’s an error, or a network timeout, etc. You should be able to, for example, safely block the interface waiting for a reply from the server. If any of the APIs don’t behave this way it should be treated as a bug.

The pending event interface consists of 2 methods:

- `getPendingEventCount()` Returns the number of pending events on the queue.
- `Variant popPendingEvent()` Pops the first event from the queue and returns it.
Store Kit

Implemented in platform/iphone/in_app_store.mm

The Store Kit API is accessible through the “InAppStore” singleton (will always be available from gdscript). It is initialized automatically. It has 2 methods for purchasing:

- Error purchase(Variant p_params);
- Error request_product_info(Variant p_params);

and the pending_event interface

```cpp
int get_pending_event_count();
Variant pop_pending_event();
```

purchase

Purchases a product id through the Store Kit API.

**Parameters**

Takes a Dictionary as a parameter, with one field, product_id, a string with your product id. Example:

```cpp
var result = InAppStore.purchase( { "product_id": "my_product" } )
```

**Response event**

The response event will be a dictionary with the following fields:

On error:

```json
{
  "type": "purchase",
  "result": "error",
  "product_id": "the product id requested"
}
```

On success:

```json
{
  "type": "purchase",
  "result": "ok",
  "product_id": "the product id requested"
}
```

request_product_info

Requests the product info on a list of product IDs.
Parameters

Takes a Dictionary as a parameter, with one field, `product_ids`, a string array with a list of product ids. Example:

```javascript
var result = InAppStore.request_product_info({ "product_ids": ["my_product1", "my_product2"] });
```

Response event

The response event will be a dictionary with the following fields:

```javascript
{
    "type": "product_info",
    "result": "ok",
    "invalid_ids": [ list of requested ids that were invalid ],
    "ids": [ list of ids that were valid ],
    "titles": [ list of valid product titles (corresponds with list of valid ids) ],
    "descriptions": [ list of valid product descriptions ],
    "prices": [ list of valid product prices ],
    "localized_prices": [ list of valid product localized prices ],
}
```

Game Center

Implemented in platform/iphone/game_center.mm

The Game Center API is available through the “GameCenter” singleton. It has 6 methods:

- `Error post_score(Variant p_score);`
- `Error award_achievement(Variant p_params);`
- `Error reset_achievements();`
- `Error request_achievements();`
- `Error request_achievement_descriptions();`
- `Error show_game_center(Variant p_params);`

plus the standard pending event interface.

**post_score**

Posts a score to a Game Center leaderboard.

Parameters

Takes a Dictionary as a parameter, with 2 fields:

- `score` a float number
- `category` a string with the category name

Example:
**Response event**

The response event will be a dictionary with the following fields:

**On error:**

```json
{
  "type": "post_score",
  "result": "error",
  "error_code": the value from NSError::code,
  "error_description": the value from NSError::localizedDescription,
}
```

**On success:**

```json
{
  "type": "post_score",
  "result": "ok",
}
```

**award_achievement**

Modifies the progress of a Game Center achievement.

**Parameters**

Takes a Dictionary as a parameter, with 3 fields:

- **name** (string) the achievement name
- **progress** (float) the achievement progress from 0.0 to 100.0 (passed to GKAchievement::percentComplete)
- **show_completion_banner** (bool) whether Game Center should display an achievement banner at the top of the screen

**Example:**

```javascript
var result = award_achievement( { "name": "hard_mode_completed", "progress": 6.1 } )
```

**Response event**

The response event will be a dictionary with the following fields:

**On error:**

```json
{
  "type": "award_achievement",
  "result": "error",
  "error_code": the error code taken from NSError::code,
}
```
On success:

```
{
    "type": "award_achievement",
    "result": "ok",
}
```

**reset_achievements**

Clears all Game Center achievements. The function takes no parameters.

**Response event**

The response event will be a dictionary with the following fields:

On error:

```
{
    "type": "reset_achievements",
    "result": "error",
    "error_code": the value from NSError::code
}
```

On success:

```
{
    "type": "reset_achievements",
    "result": "ok",
}
```

**request_achievements**

Request all the Game Center achievements the player has made progress on. The function takes no parameters.

**Response event**

The response event will be a dictionary with the following fields:

On error:

```
{
    "type": "achievements",
    "result": "error",
    "error_code": the value from NSError::code
}
```

On success:

```
{
    "type": "achievements",
    "result": "ok",
    "names": [ list of the name of each achievement ],
}
```
request_achievement_descriptions

Request the descriptions of all existing Game Center achievements regardless of progress. The function takes no parameters.

Response event

The response event will be a dictionary with the following fields:

On error:

```
{
    "type": "achievement_descriptions",
    "result": "error",
    "error_code": the value from NSError::code
}
```

On success:

```
{
    "type": "achievement_descriptions",
    "result": "ok",
    "names": [ list of the name of each achievement ],
    "titles": [ list of the title of each achievement ]
    "unachieved_descriptions": [ list of the description of each achievement when it is unachieved ]
    "achieved_descriptions": [ list of the description of each achievement when it is achieved ]
    "maximum_points": [ list of the points earned by completing each achievement ]
    "hidden": [ list of booleans indicating whether each achievement is initially visible ]
    "replayable": [ list of booleans indicating whether each achievement can be earned more than once ]
}
```

show_game_center

Displays the built in Game Center overlay showing leaderboards, achievements, and challenges.

Parameters

Takes a Dictionary as a parameter, with 2 fields:

- **view** (string) (optional) the name of the view to present. Accepts “default”, “leaderboards”, “achievements”, or “challenges”. Defaults to “default”.
- **leaderboard_name** (string) (optional) the name of the leaderboard to present. Only used when “view” is “leaderboards” (or “default” is configured to show leaderboards). If not specified, Game Center will display the aggregate leaderboard.
Examples:

```gdscript
var result = show_game_center( { "view": "leaderboards", "leaderboard_name": "best_time_leaderboard" } )
var result = show_game_center( { "view": "achievements" } )
```

Response event

The response event will be a dictionary with the following fields:

On close:

```json
{
  "type": "show_game_center",
  "result": "ok",
}
```

Multi-platform games

When working on a multi-platform game, you won’t always have the “GameCenter” singleton available (for example when running on PC or Android). Because the gdscript compiler looks up the singletons at compile time, you can’t just query the singletons to see and use what you need inside a conditional block, you need to also define them as valid identifiers (local variable or class member). This is an example of how to work around this in a class:

```gdscript
var GameCenter = null # define it as a class member

func post_score(p_score):
    if GameCenter == null:
        return
    GameCenter.post_score( { "value": p_score, "category": "my_leaderboard" } )

func check_events():
    while GameCenter.get_pending_event_count() > 0:
        # do something with events here
        pass

func _ready():
    # check if the singleton exists
    if Globals.has_singleton("GameCenter"):
        GameCenter = Globals.get_singleton("GameCenter")
        # connect your timer here to the "check_events" function
```

4.13 Miscellaneous

4.13.1 Background loading

When switching the main scene of your game (for example going to a new level), you might want to show a loading screen with some indication that progress is being made. The main load method (ResourceLoader::load or just load from gdscript) blocks your thread while the resource is being loaded, so it’s not good. This document discusses the ResourceInteractiveLoader class for smoother load screens.
**ResourceInteractiveLoader**

The `ResourceInteractiveLoader` class allows you to load a resource in stages. Every time the method `poll` is called, a new stage is loaded, and control is returned to the caller. Each stage is generally a sub-resource that is loaded by the main resource. For example, if you’re loading a scene that loads 10 images, each image will be one stage.

**Usage**

Usage is generally as follows:

**Obtaining a ResourceInteractiveLoader**

```cpp
Ref<ResourceInteractiveLoader> ResourceLoader::load_interactive(String p_path);
```

This method will give you a `ResourceInteractiveLoader` that you will use to manage the load operation.

**Polling**

```cpp
Error ResourceInteractiveLoader::poll();
```

Use this method to advance the progress of the load. Each call to `poll` will load the next stage of your resource. Keep in mind that each stage is one entire “atomic” resource, such as an image, or a mesh, so it will take several frames to load.

Returns `OK` on no errors, `ERR_FILE_EOF` when loading is finished. Any other return value means there was an error and loading has stopped.

**Load progress (optional)**

To query the progress of the load, use the following methods:

```cpp
int ResourceInteractiveLoader::get_stage_count() const;
int ResourceInteractiveLoader::get_stage() const;
```

`get_stage_count` returns the total number of stages to load, `get_stage` returns the current stage being loaded.

**Forcing completion (optional)**

```cpp
Error ResourceInteractiveLoader::wait();
```

Use this method if you need to load the entire resource in the current frame, without any more steps.

**Obtaining the resource**

```cpp
Ref<Resource> ResourceInteractiveLoader::get_resource();
```

If everything goes well, use this method to retrieve your loaded resource.
Example

This example demonstrates how to load a new scene. Consider it in the context of the Singletons (AutoLoad) example.

First we setup some variables and initialize the `current_scene` with the main scene of the game:

```godot
var loader
var wait_frames
var time_max = 100 # msec
var current_scene

func _ready():
    var root = get_tree().get_root()
    current_scene = root.get_child(root.get_child_count() -1)
```

The function `goto_scene` is called from the game when the scene needs to be switched. It requests an interactive loader, and calls `set_progress(true)` to start polling the loader in the `_progress` callback. It also starts a “loading” animation, which can show a progress bar or loading screen, etc.

```godot
func goto_scene(path):
    # game requests to switch to this scene
    loader = ResourceLoader.load_interactive(path)
    if loader == null:
        # check for errors
        show_error()
        return
    set_process(true)
    current_scene.queue_free() # get rid of the old scene
    # start your "loading..." animation
    get_node("animation").play("loading")
    wait_frames = 1
```

`_process` is where the loader is polled. `poll` is called, and then we deal with the return value from that call. OK means keep polling, ERR_FILE_EOF means load is done, anything else means there was an error. Also note we skip one frame (via `wait_frames`, set on the `goto_scene` function) to allow the loading screen to show up.

Note how use `OS.get_ticks_msec` to control how long we block the thread. Some stages might load really fast, which means we might be able to cram more than one call to `poll` in one frame, some might take way more than your value for `time_max`, so keep in mind we won’t have precise control over the timings.

```godot
func _process(time):
    if loader == null:
        # no need to process anymore
        set_process(false)
        return

    if wait_frames > 0:
        # wait for frames to let the "loading" animation to show up
        wait_frames -= 1
        return

    var t = OS.get_ticks_msec()
    while OS.get_ticks_msec() < t + time_max: # use "time_max" to control how much
        # time we block this thread

        # poll your loader
        var err = loader.poll()

        if err == ERR_FILE_EOF: # load finished
```
Some extra helper functions. `update_progress` updates a progress bar, or can also update a paused animation (the animation represents the entire load process from beginning to end). `set_new_scene` puts the newly loaded scene on the tree. Because it’s a scene being loaded, `instance()` needs to be called on the resource obtained from the loader.

```gd
func update_progress():
    var progress = float(loader.get_stage()) / loader.get_stage_count()
    # update your progress bar?
    get_node("progress").set_progress(progress)

    # or update a progress animation?
    var len = get_node("animation").get_current_animation_length()

    # call this on a paused animation. use "true" as the second parameter to force
    # the animation to update
    get_node("animation").seek(progress * len, true)

func set_new_scene(scene_resource):
    current_scene = scene_resource.instance()
    get_node("/root").add_child(current_scene)
```

### Using multiple threads

`ResourceInteractiveLoader` can be used from multiple threads. A couple of things to keep in mind if you attempt it:

#### Use a Semaphore

While your thread waits for the main thread to request a new resource, use a Semaphore to sleep (instead of a busy loop or anything similar).

#### Not blocking main thread during the polling

If you have a mutex to allow calls from the main thread to your loader class, don’t lock it while you call `poll` on the loader. When a resource is finished loading, it might require some resources from the low level APIs (VisualServer, etc), which might need to lock the main thread to acquire them. This might cause a deadlock if the main thread is waiting for your mutex while your thread is waiting to load a resource.

#### Example class

You can find an example class for loading resources in threads here: `resource_queue.gd`. Usage is as follows:
**func start()**

Call after you instance the class to start the thread.

**func queue_resource(path, p_in_front = false)**

Queue a resource. Use optional parameter “p_in_front” to put it in front of the queue.

**func cancel_resource(path)**

Remove a resource from the queue, discarding any loading done.

**func is_ready(path)**

Returns true if a resource is done loading and ready to be retrieved.

**func get_progress(path)**

Get the progress of a resource. Returns -1 on error (for example if the resource is not on the queue), or a number between 0.0 and 1.0 with the progress of the load. Use mostly for cosmetic purposes (updating progress bars, etc), use is_ready to find out if a resource is actually ready.

**func get_resource(path)**

Returns the fully loaded resource, or null on error. If the resource is not done loading (is_ready returns false), it will block your thread and finish the load. If the resource is not on the queue, it will call ResourceLoader::load to load it normally and return it.

**Example:**

```godot
define start()
define queue_resource(path, p_in_front = false)
define cancel_resource(path)
define is_ready(path)
define get_progress(path)
define get_resource(path)

# initialize
queue = preload("res://resource_queue.gd").new()
queue.start()

# suppose your game starts with a 10 second cutscene, during which the user can't interact with the game.
# For that time we know they won't use the pause menu, so we can queue it to load during the cutscene:
queue.queue_resource("res://pause_menu.xml")
start_curscene()

# later when the user presses the pause button for the first time:
pause_menu = queue.get_resource("res://pause_menu.xml").instance()
pause_menu.show()

# when you need a new scene:
queue.queue_resource("res://level_1.xml", true) # use "true" as the second parameter to put it at the front of the queue, pausing the load of any other resource

# to check progress
if queue.is_ready("res://level_1.xml"):  # use "true" as the second parameter to check if any other resource is done loading
    show_new_level(queue.get_resource("res://level_1.xml"))
else:
    update_progress(queue.get_progress("res://level_1.xml"))
```

296 Chapter 4. Engine features
# when the user walks away from the trigger zone in your Metroidvania game:
queue.cancel_resource("res://zone_2.xml")

Note: this code in its current form is not tested in real world scenarios. Ask punto on IRC (#godotengine on irc.freenode.net) for help.

## 4.13.2 Data paths

### Path separators

For the sake of supporting as many platforms as possible, Godot only accepts unix style path separators (/). These work everywhere, including Windows.

A path like: C:\Projects will become C:/Projects.

### Resource path

As mentioned before. Godot considers that a project exists at any given folder that contains an “engine.cfg” text file, even if such file is empty.

Accessing project files can be done by opening any path with res:// as a base. For example, a texture located in the root of the project folder may be opened from the following path: res://sometexture.png.

### Userdata path (persistent data)

While the project is running, it is a very common scenario that the resource path will be read-only, due to it being inside a package, self contained executable, or system wide install location.

Storing persistent files in such scenarios should be done by using the user:// prefix, for example: user://gamesave.txt.

In some devices (for example, mobile ad consoles) this path is unique for the app. Under desktop operating systems, the engine uses the typical ~/.Name (check the project name under the settings) in OSX and Linux, and APPDATA/Name for Windows.

## 4.13.3 Saving games

### Introduction

Save games can be complicated. It can be desired to store more information than the current level or number of stars earned on a level. More advanced save games may need to store additional information about an arbitrary number of objects. This will allow the save function to scale as the game grows more complex.

### Identify persistent objects

First we should identify what objects we want to keep between game sessions and what information we want to keep from those objects. For this tutorial, we will use groups to mark and handle objects to be saved but other methods are certainly possible.

We will start by adding objects we wish to save to the “Persist” group. As in the Scripting (continued) tutorial, we can do this through the GUI or through script. Let’s add the relevant nodes using the GUI:
Once this is done when we need to save the game we can get all objects to save them and then tell them all to save with this script:

```gdscript
var savenodes = get_tree().get_nodes_in_group("Persist")
for i in savenodes:
    # Now we can call our save function on each node.
```

## Serializing

The next step is to serialize the data. This makes it much easier to read and store to disk. In this case, we’re assuming each member of group Persist is an instanced node and thus has a path. GDScript has helper functions for this, such as `Dictionary.to_json()` and `Dictionary.parse_json()`, so we will use a dictionary. Our node needs to contain a save function that returns this data. The save function will look like this:

```gdscript
func save():
    var savedict = {
        filename=get_filename(),
        parent=get_parent().get_path(),
        posx=get_pos().x,  #Vector2 is not supported by json
        posy=get_pos().y,
        attack=attack,
        defense=defense,
        currenthealth=currenthealth,
        maxhealth=maxhealth,
        damage=damage,
        regen=regen,
        experience=experience,
        TNL=TNL,
        level=level,
        AttackGrowth=AttackGrowth,
        DefenseGrowth=DefenseGrowth,
        HealthGrowth=HealthGrowth,
        isalive=isalive,
        last_attack=last_attack
    }
    return savedict
```

This gives us a dictionary with the style `{ "variable_name":that_variables_value }` which will be useful when loading.

### Saving and reading data

As covered in the *File system* tutorial, we’ll need to open a file and write to it and then later read from it. Now that we have a way to call our groups and get their relevant data, let’s use to_json() to convert it into an easily stored string and store them in a file. Doing it this way ensures that each line is its own object so we have an easy way to pull the data out of the file as well.

```gdscript
# Note: This can be called from anywhere inside the tree. This function is path independent.
# Go through everything in the persist category and ask them to return a dict of relevant variables
```
func save_game():
    var savegame = File.new()
    savegame.open("user://savegame.save", File.WRITE)
    var savenodes = get_tree().get_nodes_in_group("Persist")
    for i in savenodes:
        var nodedata = i.save()
        savegame.store_line(nodedata.to_json())
    savegame.close()

Game saved! Loading is fairly simple as well. For that we‘ll read each line, use parse_json() to read it back to a dict, and then iterate over the dict to read our values. But we‘ll need to first create the object and we can use the filename and parent values to achieve that. Here is our load function:

# Note: This can be called from anywhere inside the tree. This function is path-independent.
func load_game():
    var savegame = File.new()
    if !savegame.file_exists("user://savegame.save"): return
    #Error! We don't have a save to load

    # We need to revert the game state so we're not cloning objects during loading.
    # This will vary wildly depending on the needs of a project, so take care with this step.
    # For our example, we will accomplish this by deleting savable objects.
    var savenodes = get_tree().get_nodes_in_group("Persist")
    for i in savenodes:
        i.queue_free()

    # Load the file line by line and process that dictionary to restore the object it represents
    var currentline = {} # dict.parse_json() requires a declared dict.
    savegame.open("user://savegame.save", File.READ)
    currentline.parse_json(savegame.get_line())
    while (!savegame.eof_reached()):
        # First we need to create the object and add it to the tree and set its position.
        var newobject = load(currentline["filename"]).instance()
        get_node(currentline["parent"]).add_child(newobject)
        newobject.set_pos(Vector2(currentline["posx"],currentline["posy"]))
        # Now we set the remaining variables.
        for i in currentline.keys():
            if (i == "filename" or i == "parent" or i == "posx" or i == "posy"): continue
            newobject.set(i, currentline[i])
        currentline.parse_json(savegame.get_line())
    savegame.close()

And now we can save and load an arbitrary number of objects laid out almost anywhere across the scene tree! Each object can store different data depending on what it needs to save.

Some notes

We may have glossed over a step, but setting the game state to one fit to start loading data can be very complicated. This step will need to be heavily customized based on the needs of an individual project.

This implementation assumes no Persist objects are children of other Persist objects. Doing so would create invalid paths. If this is one of the needs of a project this needs to be considered. Saving objects in stages (parent objects first)

4.13. Miscellaneous 299
so they are available when child objects are loaded will make sure they’re available for the add_child() call. There will also need to be some way to link children to parents as the nodepath will likely be invalid.

4.13.4 Encrypting save games

Why?

Because the world today is not the world of yesterday. A capitalist oligarchy runs the world and forces us to consume in order to keep the gears of this rotten society on track. As such, the biggest market for video game consumption today is the mobile one. It is a market of poor souls forced to compulsively consume digital content in order to forget the misery of their every day life, commute, or just any other brief free moment they have that they are not using to produce goods or services for the ruling class. These individuals need to keep focusing on their video games (because not doing so will produce them a tremendous existential angst), so they go as far as spending money on them to extend their experience, and their preferred way of doing so is through in-app purchases and virtual currency.

But, imagine if someone was to find a way to edit the saved games and assign the items and currency without effort? This would be terrible, because it would help players consume the content much faster, and as such run out of it sooner than expected. If this happens they will have nothing that avoids them to think, and the tremendous agony of realizing their own irrelevance would again take over their life.

No, we definitely do not want this to happen, so let’s see how to encrypt savegames and protect the world order.

How?

The class File is simple to use, just open a location and read/write data (integers, strings and variants). To create an encrypted file, a passphrase must be provided, like this:

```javascript
var f = File.new()
var err = f.open_encrypted_with_pass("user://savedata.bin", File.WRITE, "mypass")
f.store_var(game_state)
f.close()
```

This will make the file unreadable to users, but will still not avoid them to share savefiles. To solve this, using the device unique id or some unique user identifier is needed, for example:

```javascript
var f = File.new()
var err = f.open_encrypted_with_pass("user://savedata.bin", File.WRITE, OS.get_unique_ID())
f.store_var(game_state)
f.close()
```

Note that OS.get_unique_ID() only works on iOS and Android.

This is all! Thanks for your cooperation, citizen.

4.13.5 Handling quit requests

Quitting

Most platforms have the option to request the application to quit. On desktops, this is usually done with the “x” icon on the window titlebar. On Android, the back button is used to quit when on the main screen (and to go back otherwise).
Handling the notification

The MainLoop has a special notification that is sent to all nodes when quit is requested: MainLoop.NOTIFICATION_WM_QUIT.

Handling it is done as follows (on any node):

```python
func _notification(what):
    if (what == MainLoop.NOTIFICATION_WM_QUIT_REQUEST):
        get_tree().quit()  # default behavior
```

When developing mobile apps, quitting is not desired unless the user is on the main screen, so the behavior can be changed.

It is important to note that by default, Godot apps have the built-in behavior to quit when quit is requested, this can be changed:

```python
get_tree().set_auto_accept_quit(false)
```

4.13.6 Pausing games

Pause?

In most games it is desirable to, at some point, interrupt the game to do something else, such as taking a break or changing options. However this is not as simple as it seems. The game might be stopped, but it might be desirable that some menus and animations continue working.

Implementing a fine-grained control for what can be paused (and what can not) is a lot of work, so a simple framework for pausing is provided in Godot.

How pausing works

To set pause mode, the pause state must be set. This is done by calling `SceneTree.set_pause()` with a “true” argument:

```python
get_tree().set_pause(true)
```

Doing so will have the following behavior:

- 2D and 3D physics will be stopped.
- _process and _fixed_process will not be called anymore in nodes.
- _input and _input_event will not be called anymore either.

This effectively stops the whole game. Calling this function from a script, by default, will result in an unrecoverable state (nothing will work anymore!).

White-listing nodes

Before enabling pause, make sure that nodes that must keep working during pause are white-listed. This is done by editing the “Pause Mode” property in a node:
By default all nodes have this property in the “Inherit” state. This means, that they will only process (or not) depending on what this same property is set on the parent node. If the parent is set to “Inherit”, then the grandparent will be checked and so on. Ultimately, if a state can’t be found in any of the grandparents, the pause state in SceneTree is used. This means that, by default, when the game is paused every node will be paused.

So the three possible states for a node are:

- **Inherit**: Process depending on the state of the parent, grandparent, etc. The first parent that has a non-Inherit state.
- **Stop**: Stop the node no matter what (and children in Inherit mode). When paused this node will not process.
- **Process**: Process the node no matter what (and children in Inherit mode). Paused or not this node will process.

**Example**

An example of this is creating a popup or panel with controls inside, and set its pause mode to “Process” then just hide it:
Just by setting the root of the pause popup to “Process”, all children and grandchildren will inherit that state. This way, this branch of the scene tree will continue working when paused.

Finally, make it so when a pause button is pressed (any button will do), enable the pause and show the pause screen.

```godot
func _on_pause_button_pressed():
    get_tree().set_pause(true)
    get_node("pause_popup").show()
```

To remove the pause, just do the opposite when the pause screen is closed:

```godot
func _on_pause_popup_close_pressed():
    get_node("pause_popup").hide()
    get_tree().set_pause(false)
```

And that should be all!

### 4.13.7 Internationalizing games

#### Introduction

Sería excelente que el mundo hablara solo un idioma. Unfortunately for us developers, that is not the case. While not generally a big requirement when developing indie or niche games, it is also very common that games going into a more massive market require localization.

Godot offers many tools to make this process more straightforward, so this tutorial is more like a collection of tips and tricks.

Localization is usually done by specific studios hired for the job and, despite the huge amount of software and file formats available for this, the most common way to do localization to this day is still with spreadsheets. The process of creating the spreadsheets and importing them is already covered in the *Importing translations* tutorial, so this one could be seen more like a follow up to that one.

#### Configuring the imported translation

The translations can get updated and re-imported when they change, but they still have to be added to the project. This is done in Scene > Project Settings > Localization:
This dialog allows to add or remove translations project-wide.

**Localizing resources**

It is also possible to instruct Godot to open alternative versions of assets (resources) depending on the current language. For this the “Remaps” tab exists:

Select the resource to be remapped, and the alternatives for each locale.

**Converting keys to text**

Some controls such as *Button*, *Label*, etc. will automatically fetch a translation each time they are set a key instead of a text. For example, if a label is assigned “MAIN_SCREEN_GREETING1” and a key to different languages exists in the translations, this will be automatically converted. This process is done upon load though, so if the project in
question has a dialog that allows changing the language in the settings, the scenes (or at least the settings scene) will have to be re-loaded for new text to have effect.

For code, the `Object.tr()` function can be used. This will just look-up the text into the translations and convert it if found:

```python
level.set_text(tr("LEVEL_5_NAME"))
status.set_text(tr("GAME_STATUS_" + str(status_index)))
```

Making controls resizeable

The same text in different languages can vary greatly in length. For this, make sure to read the tutorial on Size and anchors, as having dynamically adjusted control sizes may help. Container can be very useful, as well as the multiple options in Label for text wrapping.

TranslationServer

Godot has a server for handling the low level translation management called the TranslationServer. Translations can be added or removed during run-time, and the current language be changed too.

Command line

Language can be tested when running Godot from command line. For example, to test a game in french, the following arguments can be supplied:

```
c:\MyGame> godot -lang fr
```

Translating the project name

The project name becomes the app name when exporting to different operating systems and platforms. To specify the project name in more than one language, create a new setting application/name in the project settings dialog and append the locale identifier to it. For example:
4.13.8 Locales

This is the list of supported locales and variants in the engine. It's based on the Unix standard locale strings:

<table>
<thead>
<tr>
<th>Locale</th>
<th>Language and Variant</th>
</tr>
</thead>
<tbody>
<tr>
<td>ar</td>
<td>Arabic</td>
</tr>
<tr>
<td>ar_AE</td>
<td>Arabic (United Arab Emirates)</td>
</tr>
<tr>
<td>ar_BH</td>
<td>Arabic (Bahrain)</td>
</tr>
<tr>
<td>ar_DZ</td>
<td>Arabic (Algeria)</td>
</tr>
<tr>
<td>ar_EG</td>
<td>Arabic (Egypt)</td>
</tr>
<tr>
<td>ar_IQ</td>
<td>Arabic (Iraq)</td>
</tr>
<tr>
<td>ar_JO</td>
<td>Arabic (Jordan)</td>
</tr>
<tr>
<td>ar_KW</td>
<td>Arabic (Kuwait)</td>
</tr>
<tr>
<td>ar_LB</td>
<td>Arabic (Lebanon)</td>
</tr>
<tr>
<td>ar_LY</td>
<td>Arabic (Libya)</td>
</tr>
<tr>
<td>ar_MA</td>
<td>Arabic (Morocco)</td>
</tr>
<tr>
<td>ar_OM</td>
<td>Arabic (Oman)</td>
</tr>
<tr>
<td>ar_QA</td>
<td>Arabic (Qatar)</td>
</tr>
</tbody>
</table>

As always, If you don’t know the code of a language or zone, check the list.
Table 4.2 – continued from previous page

<table>
<thead>
<tr>
<th>Locale</th>
<th>Language and Variant</th>
</tr>
</thead>
<tbody>
<tr>
<td>ar_SA</td>
<td>Arabic (Saudi Arabia)</td>
</tr>
<tr>
<td>ar_SD</td>
<td>Arabic (Sudan)</td>
</tr>
<tr>
<td>ar_SY</td>
<td>Arabic (Syria)</td>
</tr>
<tr>
<td>ar_TN</td>
<td>Arabic (Tunisia)</td>
</tr>
<tr>
<td>ar_YE</td>
<td>Arabic (Yemen)</td>
</tr>
<tr>
<td>be</td>
<td>Belarusian</td>
</tr>
<tr>
<td>be_BY</td>
<td>Belarusian (Belarus)</td>
</tr>
<tr>
<td>bg</td>
<td>Bulgarian</td>
</tr>
<tr>
<td>bg_BG</td>
<td>Bulgarian (Bulgaria)</td>
</tr>
<tr>
<td>ca</td>
<td>Catalan</td>
</tr>
<tr>
<td>ca_ES</td>
<td>Catalan (Spain)</td>
</tr>
<tr>
<td>cs</td>
<td>Czech</td>
</tr>
<tr>
<td>cs_CZ</td>
<td>Czech (Czech Republic)</td>
</tr>
<tr>
<td>da</td>
<td>Danish</td>
</tr>
<tr>
<td>da_DK</td>
<td>Danish (Denmark)</td>
</tr>
<tr>
<td>de</td>
<td>German</td>
</tr>
<tr>
<td>de_AT</td>
<td>German (Austria)</td>
</tr>
<tr>
<td>de_CH</td>
<td>German (Switzerland)</td>
</tr>
<tr>
<td>de_DE</td>
<td>German (Germany)</td>
</tr>
<tr>
<td>de_LU</td>
<td>German (Luxembourg)</td>
</tr>
<tr>
<td>el</td>
<td>Greek</td>
</tr>
<tr>
<td>el_CY</td>
<td>Greek (Cyprus)</td>
</tr>
<tr>
<td>el_GR</td>
<td>Greek (Greece)</td>
</tr>
<tr>
<td>en</td>
<td>English</td>
</tr>
<tr>
<td>en_AU</td>
<td>English (Australia)</td>
</tr>
<tr>
<td>en_CA</td>
<td>English (Canada)</td>
</tr>
<tr>
<td>en_GB</td>
<td>English (United Kingdom)</td>
</tr>
<tr>
<td>en_IE</td>
<td>English (Ireland)</td>
</tr>
<tr>
<td>en_IN</td>
<td>English (India)</td>
</tr>
<tr>
<td>en_MT</td>
<td>English (Malta)</td>
</tr>
<tr>
<td>en_NZ</td>
<td>English (New Zealand)</td>
</tr>
<tr>
<td>en_PH</td>
<td>English (Philippines)</td>
</tr>
<tr>
<td>en_SG</td>
<td>English (Singapore)</td>
</tr>
<tr>
<td>en_US</td>
<td>English (United States)</td>
</tr>
<tr>
<td>en_ZA</td>
<td>English (South Africa)</td>
</tr>
<tr>
<td>es</td>
<td>Spanish</td>
</tr>
<tr>
<td>es_AR</td>
<td>Spanish (Argentina)</td>
</tr>
<tr>
<td>es_BO</td>
<td>Spanish (Bolivia)</td>
</tr>
<tr>
<td>es_CL</td>
<td>Spanish (Chile)</td>
</tr>
<tr>
<td>es_CO</td>
<td>Spanish (Colombia)</td>
</tr>
<tr>
<td>es_CR</td>
<td>Spanish (Costa Rica)</td>
</tr>
<tr>
<td>es_DO</td>
<td>Spanish (Dominican Republic)</td>
</tr>
<tr>
<td>es_EC</td>
<td>Spanish (Ecuador)</td>
</tr>
<tr>
<td>es_ES</td>
<td>Spanish (Spain)</td>
</tr>
<tr>
<td>es_GT</td>
<td>Spanish (Guatemala)</td>
</tr>
<tr>
<td>es_HN</td>
<td>Spanish (Honduras)</td>
</tr>
<tr>
<td>es_MX</td>
<td>Spanish (Mexico)</td>
</tr>
<tr>
<td>es_NI</td>
<td>Spanish (Nicaragua)</td>
</tr>
<tr>
<td>es_PA</td>
<td>Spanish (Panama)</td>
</tr>
</tbody>
</table>

Continued on next page
Table 4.2 – continued from previous page

<table>
<thead>
<tr>
<th>Locale</th>
<th>Language and Variant</th>
</tr>
</thead>
<tbody>
<tr>
<td>es_PE</td>
<td>Spanish (Peru)</td>
</tr>
<tr>
<td>es_PR</td>
<td>Spanish (Puerto Rico)</td>
</tr>
<tr>
<td>es_PY</td>
<td>Spanish (Paraguay)</td>
</tr>
<tr>
<td>es_SV</td>
<td>Spanish (El Salvador)</td>
</tr>
<tr>
<td>es_US</td>
<td>Spanish (United States)</td>
</tr>
<tr>
<td>es_UY</td>
<td>Spanish (Uruguay)</td>
</tr>
<tr>
<td>es_VE</td>
<td>Spanish (Venezuela)</td>
</tr>
<tr>
<td>et</td>
<td>Estonian</td>
</tr>
<tr>
<td>et_EE</td>
<td>Estonian (Estonia)</td>
</tr>
<tr>
<td>fi</td>
<td>Finnish</td>
</tr>
<tr>
<td>fi_FI</td>
<td>Finnish (Finland)</td>
</tr>
<tr>
<td>fr</td>
<td>French</td>
</tr>
<tr>
<td>fr_BE</td>
<td>French (Belgium)</td>
</tr>
<tr>
<td>fr_CA</td>
<td>French (Canada)</td>
</tr>
<tr>
<td>fr_CH</td>
<td>French (Switzerland)</td>
</tr>
<tr>
<td>fr_FR</td>
<td>French (France)</td>
</tr>
<tr>
<td>fr_LU</td>
<td>French (Luxembourg)</td>
</tr>
<tr>
<td>ga</td>
<td>Irish</td>
</tr>
<tr>
<td>ga_IE</td>
<td>Irish (Ireland)</td>
</tr>
<tr>
<td>hi</td>
<td>Hindi (India)</td>
</tr>
<tr>
<td>hi_IN</td>
<td>Hindi (India)</td>
</tr>
<tr>
<td>hr</td>
<td>Croatian</td>
</tr>
<tr>
<td>hr_HR</td>
<td>Croatian (Croatia)</td>
</tr>
<tr>
<td>hu</td>
<td>Hungarian</td>
</tr>
<tr>
<td>hu_HU</td>
<td>Hungarian (Hungary)</td>
</tr>
<tr>
<td>in</td>
<td>Indonesian</td>
</tr>
<tr>
<td>in_ID</td>
<td>Indonesian (Indonesia)</td>
</tr>
<tr>
<td>is</td>
<td>Icelandic</td>
</tr>
<tr>
<td>is_IS</td>
<td>Icelandic (Iceland)</td>
</tr>
<tr>
<td>it</td>
<td>Italian</td>
</tr>
<tr>
<td>it_CH</td>
<td>Italian (Switzerland)</td>
</tr>
<tr>
<td>it_IT</td>
<td>Italian (Italy)</td>
</tr>
<tr>
<td>iw</td>
<td>Hebrew</td>
</tr>
<tr>
<td>iw_IL</td>
<td>Hebrew (Israel)</td>
</tr>
<tr>
<td>ja</td>
<td>Japanese</td>
</tr>
<tr>
<td>ja_JP</td>
<td>Japanese (Japan)</td>
</tr>
<tr>
<td>ko</td>
<td>Korean</td>
</tr>
<tr>
<td>ko_KR</td>
<td>Korean (South Korea)</td>
</tr>
<tr>
<td>lt</td>
<td>Lithuanian</td>
</tr>
<tr>
<td>lt_LT</td>
<td>Lithuanian (Lithuania)</td>
</tr>
<tr>
<td>lv</td>
<td>Latvian</td>
</tr>
<tr>
<td>lv_LV</td>
<td>Latvian (Latvia)</td>
</tr>
<tr>
<td>mk</td>
<td>Macedonian</td>
</tr>
<tr>
<td>mk_MK</td>
<td>Macedonian (Macedonia)</td>
</tr>
<tr>
<td>ms</td>
<td>Malay</td>
</tr>
<tr>
<td>ms_MY</td>
<td>Malay (Malaysia)</td>
</tr>
<tr>
<td>mt</td>
<td>Maltese</td>
</tr>
<tr>
<td>mt_MT</td>
<td>Maltese (Malta)</td>
</tr>
</tbody>
</table>

Continued on next page
### 4.13.9 Binary serialization API

#### Introduction

Godot has a simple serialization API based on Variant. It’s used for converting data types to an array of bytes efficiently. This API is used in the functions `get_var` and `store_var` of `File` as well as the packet APIs for `PacketPeer`. This

<table>
<thead>
<tr>
<th>Locale</th>
<th>Language and Variant</th>
</tr>
</thead>
<tbody>
<tr>
<td>nl</td>
<td>Dutch</td>
</tr>
<tr>
<td>nl_BE</td>
<td>Dutch (Belgium)</td>
</tr>
<tr>
<td>nl_NL</td>
<td>Dutch (Netherlands)</td>
</tr>
<tr>
<td>no</td>
<td>Norwegian</td>
</tr>
<tr>
<td>no_NO</td>
<td>Norwegian (Norway)</td>
</tr>
<tr>
<td>no_NO_NY</td>
<td>Norwegian (Norway, Nynorsk)</td>
</tr>
<tr>
<td>pl</td>
<td>Polish</td>
</tr>
<tr>
<td>pl_PL</td>
<td>Polish (Poland)</td>
</tr>
<tr>
<td>pt</td>
<td>Portuguese</td>
</tr>
<tr>
<td>pt_BR</td>
<td>Portuguese (Brazil)</td>
</tr>
<tr>
<td>pt_PT</td>
<td>Portuguese (Portugal)</td>
</tr>
<tr>
<td>ro</td>
<td>Romanian</td>
</tr>
<tr>
<td>ro_RO</td>
<td>Romanian (Romania)</td>
</tr>
<tr>
<td>ru</td>
<td>Russian</td>
</tr>
<tr>
<td>ru_RU</td>
<td>Russian (Russia)</td>
</tr>
<tr>
<td>sk</td>
<td>Slovak</td>
</tr>
<tr>
<td>sk_SK</td>
<td>Slovak (Slovakia)</td>
</tr>
<tr>
<td>sl</td>
<td>Slovenian</td>
</tr>
<tr>
<td>sl_SI</td>
<td>Slovenian (Slovenia)</td>
</tr>
<tr>
<td>sq</td>
<td>Albanian</td>
</tr>
<tr>
<td>sq_AL</td>
<td>Albanian (Albania)</td>
</tr>
<tr>
<td>sr</td>
<td>Serbian</td>
</tr>
<tr>
<td>sr_BA</td>
<td>Serbian (Bosnia and Herzegovina)</td>
</tr>
<tr>
<td>sr_CS</td>
<td>Serbian (Serbia and Montenegro)</td>
</tr>
<tr>
<td>sr_ME</td>
<td>Serbian (Montenegro)</td>
</tr>
<tr>
<td>sr_RS</td>
<td>Serbian (Serbia)</td>
</tr>
<tr>
<td>sv</td>
<td>Swedish</td>
</tr>
<tr>
<td>sv_SE</td>
<td>Swedish (Sweden)</td>
</tr>
<tr>
<td>th</td>
<td>Thai</td>
</tr>
<tr>
<td>th_TH</td>
<td>Thai (Thailand)</td>
</tr>
<tr>
<td>th_TH_TH</td>
<td>Thai (Thailand, TH)</td>
</tr>
<tr>
<td>tr</td>
<td>Turkish</td>
</tr>
<tr>
<td>tr_TR</td>
<td>Turkish (Turkey)</td>
</tr>
<tr>
<td>uk</td>
<td>Ukrainian</td>
</tr>
<tr>
<td>uk_UA</td>
<td>Ukrainian (Ukraine)</td>
</tr>
<tr>
<td>vi</td>
<td>Vietnamese</td>
</tr>
<tr>
<td>vi_VN</td>
<td>Vietnamese (Vietnam)</td>
</tr>
<tr>
<td>zh</td>
<td>Chinese</td>
</tr>
<tr>
<td>zh_CN</td>
<td>Chinese (China)</td>
</tr>
<tr>
<td>zh_HK</td>
<td>Chinese (Hong Kong)</td>
</tr>
<tr>
<td>zh_SG</td>
<td>Chinese (Singapore)</td>
</tr>
<tr>
<td>zh_TW</td>
<td>Chinese (Taiwan)</td>
</tr>
</tbody>
</table>

---

310 Chapter 4. Engine features
format is not used for binary scenes and resources.

**Packet specification**

The packet is designed to be always padded to 4 bytes. All values are little endian encoded. All packets have a 4 byte header representing an integer, specifying the type of data:

<table>
<thead>
<tr>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>null</td>
</tr>
<tr>
<td>1</td>
<td>bool</td>
</tr>
<tr>
<td>2</td>
<td>integer</td>
</tr>
<tr>
<td>3</td>
<td>float</td>
</tr>
<tr>
<td>4</td>
<td>string</td>
</tr>
<tr>
<td>5</td>
<td>vector2</td>
</tr>
<tr>
<td>6</td>
<td>rect2</td>
</tr>
<tr>
<td>7</td>
<td>vector3</td>
</tr>
<tr>
<td>8</td>
<td>matrix32</td>
</tr>
<tr>
<td>9</td>
<td>plane</td>
</tr>
<tr>
<td>10</td>
<td>quaternion</td>
</tr>
<tr>
<td>11</td>
<td>aabb (rect3)</td>
</tr>
<tr>
<td>12</td>
<td>matrix3x3</td>
</tr>
<tr>
<td>13</td>
<td>transform (matrix 4x3)</td>
</tr>
<tr>
<td>14</td>
<td>color</td>
</tr>
<tr>
<td>15</td>
<td>image</td>
</tr>
<tr>
<td>16</td>
<td>node path</td>
</tr>
<tr>
<td>17</td>
<td>rid (unsupported)</td>
</tr>
<tr>
<td>18</td>
<td>object (unsupported)</td>
</tr>
<tr>
<td>19</td>
<td>input event</td>
</tr>
<tr>
<td>20</td>
<td>dictionary</td>
</tr>
<tr>
<td>21</td>
<td>array</td>
</tr>
<tr>
<td>22</td>
<td>ByteArray</td>
</tr>
<tr>
<td>23</td>
<td>IntArray</td>
</tr>
<tr>
<td>24</td>
<td>FloatArray</td>
</tr>
<tr>
<td>25</td>
<td>StringArray</td>
</tr>
<tr>
<td>26</td>
<td>Vector2Array</td>
</tr>
<tr>
<td>27</td>
<td>Vector3Array</td>
</tr>
<tr>
<td>28</td>
<td>ColorArray</td>
</tr>
</tbody>
</table>

Following this is the actual packet contents, which varies for each type of packet:

**0: null**

**1: bool**

<table>
<thead>
<tr>
<th>Offset</th>
<th>Len</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>4</td>
<td>Integer</td>
<td>0 for False, 1 for True</td>
</tr>
</tbody>
</table>

**2: integer**

<table>
<thead>
<tr>
<th>Offset</th>
<th>Len</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>4</td>
<td>Integer</td>
<td>Signed, 32-Bit Integer</td>
</tr>
</tbody>
</table>
### 3: float

<table>
<thead>
<tr>
<th>Offset</th>
<th>Len</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>4</td>
<td>Float</td>
<td>IEE 754 32-Bits Float</td>
</tr>
</tbody>
</table>

### 4: string

<table>
<thead>
<tr>
<th>Offset</th>
<th>Len</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>4</td>
<td>Integer</td>
<td>String Length (in Bytes)</td>
</tr>
<tr>
<td>8</td>
<td>X</td>
<td>Bytes</td>
<td>UTF-8 Encoded String</td>
</tr>
</tbody>
</table>

This field is padded to 4 bytes.

### 5: vector2

<table>
<thead>
<tr>
<th>Offset</th>
<th>Len</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>4</td>
<td>Float</td>
<td>X Coordinate</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>Float</td>
<td>Y Coordinate</td>
</tr>
</tbody>
</table>

### 6: rect2

<table>
<thead>
<tr>
<th>Offset</th>
<th>Len</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>4</td>
<td>Float</td>
<td>X Coordinate</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>Float</td>
<td>Y Coordinate</td>
</tr>
<tr>
<td>12</td>
<td>4</td>
<td>Float</td>
<td>X Size</td>
</tr>
<tr>
<td>16</td>
<td>4</td>
<td>Float</td>
<td>Y Size</td>
</tr>
</tbody>
</table>

### 7: vector3

<table>
<thead>
<tr>
<th>Offset</th>
<th>Len</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>4</td>
<td>Float</td>
<td>X Coordinate</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>Float</td>
<td>Y Coordinate</td>
</tr>
<tr>
<td>12</td>
<td>4</td>
<td>Float</td>
<td>Z Coordinate</td>
</tr>
</tbody>
</table>

### 8: matrix32

<table>
<thead>
<tr>
<th>Offset</th>
<th>Len</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>4</td>
<td>Float</td>
<td>[0][0]</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>Float</td>
<td>[0][1]</td>
</tr>
<tr>
<td>12</td>
<td>4</td>
<td>Float</td>
<td>[1][0]</td>
</tr>
<tr>
<td>16</td>
<td>4</td>
<td>Float</td>
<td>[1][1]</td>
</tr>
<tr>
<td>20</td>
<td>4</td>
<td>Float</td>
<td>[2][0]</td>
</tr>
<tr>
<td>24</td>
<td>4</td>
<td>Float</td>
<td>[2][1]</td>
</tr>
</tbody>
</table>
9: plane

<table>
<thead>
<tr>
<th>Offset</th>
<th>Len</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>4</td>
<td>Float</td>
<td>Normal X</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>Float</td>
<td>Normal Y</td>
</tr>
<tr>
<td>12</td>
<td>4</td>
<td>Float</td>
<td>Normal Z</td>
</tr>
<tr>
<td>16</td>
<td>4</td>
<td>Float</td>
<td>Distance</td>
</tr>
</tbody>
</table>

10: quaternion

<table>
<thead>
<tr>
<th>Offset</th>
<th>Len</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>4</td>
<td>Float</td>
<td>Imaginary X</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>Float</td>
<td>Imaginary Y</td>
</tr>
<tr>
<td>12</td>
<td>4</td>
<td>Float</td>
<td>Imaginary Z</td>
</tr>
<tr>
<td>16</td>
<td>4</td>
<td>Float</td>
<td>Real W</td>
</tr>
</tbody>
</table>

11: aabb (rect3)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Len</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>4</td>
<td>Float</td>
<td>X Coordinate</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>Float</td>
<td>Y Coordinate</td>
</tr>
<tr>
<td>12</td>
<td>4</td>
<td>Float</td>
<td>Z Coordinate</td>
</tr>
<tr>
<td>16</td>
<td>4</td>
<td>Float</td>
<td>X Size</td>
</tr>
<tr>
<td>20</td>
<td>4</td>
<td>Float</td>
<td>Y Size</td>
</tr>
<tr>
<td>24</td>
<td>4</td>
<td>Float</td>
<td>Z Size</td>
</tr>
</tbody>
</table>

12: matrix3x3

<table>
<thead>
<tr>
<th>Offset</th>
<th>Len</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>4</td>
<td>Float</td>
<td>[0][0]</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>Float</td>
<td>[0][1]</td>
</tr>
<tr>
<td>12</td>
<td>4</td>
<td>Float</td>
<td>[0][2]</td>
</tr>
<tr>
<td>16</td>
<td>4</td>
<td>Float</td>
<td>[1][0]</td>
</tr>
<tr>
<td>20</td>
<td>4</td>
<td>Float</td>
<td>[1][1]</td>
</tr>
<tr>
<td>24</td>
<td>4</td>
<td>Float</td>
<td>[1][2]</td>
</tr>
<tr>
<td>28</td>
<td>4</td>
<td>Float</td>
<td>[2][0]</td>
</tr>
<tr>
<td>32</td>
<td>4</td>
<td>Float</td>
<td>[2][1]</td>
</tr>
<tr>
<td>36</td>
<td>4</td>
<td>Float</td>
<td>[2][2]</td>
</tr>
</tbody>
</table>
### 13: transform (matrix 4x3)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Len</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>4</td>
<td>Float</td>
<td>[0][0]</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>Float</td>
<td>[0][1]</td>
</tr>
<tr>
<td>12</td>
<td>4</td>
<td>Float</td>
<td>[0][2]</td>
</tr>
<tr>
<td>16</td>
<td>4</td>
<td>Float</td>
<td>[1][0]</td>
</tr>
<tr>
<td>20</td>
<td>4</td>
<td>Float</td>
<td>[1][1]</td>
</tr>
<tr>
<td>24</td>
<td>4</td>
<td>Float</td>
<td>[1][2]</td>
</tr>
<tr>
<td>28</td>
<td>4</td>
<td>Float</td>
<td>[2][0]</td>
</tr>
<tr>
<td>32</td>
<td>4</td>
<td>Float</td>
<td>[2][1]</td>
</tr>
<tr>
<td>36</td>
<td>4</td>
<td>Float</td>
<td>[2][2]</td>
</tr>
<tr>
<td>40</td>
<td>4</td>
<td>Float</td>
<td>[3][0]</td>
</tr>
<tr>
<td>44</td>
<td>4</td>
<td>Float</td>
<td>[3][1]</td>
</tr>
<tr>
<td>48</td>
<td>4</td>
<td>Float</td>
<td>[3][2]</td>
</tr>
</tbody>
</table>

### 14: color

<table>
<thead>
<tr>
<th>Offset</th>
<th>Len</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>4</td>
<td>Float</td>
<td>Red (0..1)</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>Float</td>
<td>Green (0..1)</td>
</tr>
<tr>
<td>12</td>
<td>4</td>
<td>Float</td>
<td>Blue (0..1)</td>
</tr>
<tr>
<td>16</td>
<td>4</td>
<td>Float</td>
<td>Alpha (0..1)</td>
</tr>
</tbody>
</table>

### 15: image

<table>
<thead>
<tr>
<th>Offset</th>
<th>Len</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>4</td>
<td>Integer</td>
<td>Format (see FORMAT_* in “Image”:class_image</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>Integer</td>
<td>Mip-Maps (0 means no mip-maps).</td>
</tr>
<tr>
<td>12</td>
<td>4</td>
<td>Integer</td>
<td>Width (Pixels)</td>
</tr>
<tr>
<td>16</td>
<td>4</td>
<td>Integer</td>
<td>Height (Pixels)</td>
</tr>
<tr>
<td>20</td>
<td>4</td>
<td>Integer</td>
<td>Data Length</td>
</tr>
<tr>
<td>24..24+DataLength</td>
<td>1</td>
<td>Byte</td>
<td>Image Data</td>
</tr>
</tbody>
</table>

This field is padded to 4 bytes.

### 16: node path

<table>
<thead>
<tr>
<th>Offset</th>
<th>Len</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>4</td>
<td>Integer</td>
<td>String Length, or New Format (val&amp;0x80000000!=0 and NameCount=val&amp;0x7FFFFFFF)</td>
</tr>
</tbody>
</table>

For old format:

<table>
<thead>
<tr>
<th>Offset</th>
<th>Len</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>X</td>
<td>Bytes</td>
<td>UTF-8 Encoded String</td>
</tr>
</tbody>
</table>

Padded to 4 bytes.
For new format:

<table>
<thead>
<tr>
<th>Offset</th>
<th>Len</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>4</td>
<td>Integer</td>
<td>Sub-Name Count</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>Integer</td>
<td>Flags (absolute: val&amp;1 != 0)</td>
</tr>
</tbody>
</table>

For each Name and Sub-Name

<table>
<thead>
<tr>
<th>Offset</th>
<th>Len</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X+0</td>
<td>4</td>
<td>Integer</td>
<td>String Length</td>
</tr>
<tr>
<td>X+4</td>
<td>X</td>
<td>Bytes</td>
<td>UTF-8 Encoded String</td>
</tr>
</tbody>
</table>

Every name string is padded to 4 bytes.

17: rid (unsupported)

18: object (unsupported)

19: input event

20: dictionary

<table>
<thead>
<tr>
<th>Offset</th>
<th>Len</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>4</td>
<td>Integer</td>
<td>val&amp;0x7FFFFFFF = elements, val&amp;0x80000000 = shared (bool)</td>
</tr>
</tbody>
</table>

Then what follows is, for amount of “elements”, pairs of key and value, one after the other, using this same format.

21: array

<table>
<thead>
<tr>
<th>Offset</th>
<th>Len</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>4</td>
<td>Integer</td>
<td>val&amp;0x7FFFFFFF = elements, val&amp;0x80000000 = shared (bool)</td>
</tr>
</tbody>
</table>

Then what follows is, for amount of “elements”, values one after the other, using this same format.

22: class_ByteArray

<table>
<thead>
<tr>
<th>Offset</th>
<th>Len</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>4</td>
<td>Integer</td>
<td>Array Length (Bytes)</td>
</tr>
<tr>
<td>8..8+length</td>
<td>1</td>
<td>Byte</td>
<td>Byte (0..255)</td>
</tr>
</tbody>
</table>

The array data is padded to 4 bytes.

23: IntArray

<table>
<thead>
<tr>
<th>Offset</th>
<th>Len</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>4</td>
<td>Integer</td>
<td>Array Length (Integers)</td>
</tr>
<tr>
<td>8..8+length*4</td>
<td>4</td>
<td>Integer</td>
<td>32 Bits Signed Integer</td>
</tr>
</tbody>
</table>
24: class_FloatArray

<table>
<thead>
<tr>
<th>Offset</th>
<th>Len</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>4</td>
<td>Integer</td>
<td>Array Length (Floats)</td>
</tr>
<tr>
<td>8..8+length*4</td>
<td>4</td>
<td>Integer</td>
<td>32 Bits IEE 754 Float</td>
</tr>
</tbody>
</table>

25: StringArray

<table>
<thead>
<tr>
<th>Offset</th>
<th>Len</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>4</td>
<td>Integer</td>
<td>Array Length (Strings)</td>
</tr>
</tbody>
</table>

For each String:

<table>
<thead>
<tr>
<th>Offset</th>
<th>Len</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X+0</td>
<td>4</td>
<td>Integer</td>
<td>String Length</td>
</tr>
<tr>
<td>X+4</td>
<td>X</td>
<td>Bytes</td>
<td>UTF-8 Encoded String</td>
</tr>
</tbody>
</table>

Every string is is padded to 4 bytes.

26: Vector2Array

<table>
<thead>
<tr>
<th>Offset</th>
<th>Len</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>4</td>
<td>Integer</td>
<td>Array Length</td>
</tr>
<tr>
<td>8..8+length*8</td>
<td>4</td>
<td>Float</td>
<td>X Coordinate</td>
</tr>
<tr>
<td>8..12+length*8</td>
<td>4</td>
<td>Float</td>
<td>Y Coordinate</td>
</tr>
</tbody>
</table>

27: Vector3Array

<table>
<thead>
<tr>
<th>Offset</th>
<th>Len</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>4</td>
<td>Integer</td>
<td>Array Length</td>
</tr>
<tr>
<td>8..8+length*12</td>
<td>4</td>
<td>Float</td>
<td>X Coordinate</td>
</tr>
<tr>
<td>8..12+length*12</td>
<td>4</td>
<td>Float</td>
<td>Y Coordinate</td>
</tr>
<tr>
<td>8..16+length*12</td>
<td>4</td>
<td>Float</td>
<td>Z Coordinate</td>
</tr>
</tbody>
</table>

28: ColorArray

<table>
<thead>
<tr>
<th>Offset</th>
<th>Len</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>4</td>
<td>Integer</td>
<td>Array Length</td>
</tr>
<tr>
<td>8..8+length*16</td>
<td>4</td>
<td>Float</td>
<td>Red (0..1)</td>
</tr>
<tr>
<td>8..12+length*16</td>
<td>4</td>
<td>Float</td>
<td>Green (0..1)</td>
</tr>
<tr>
<td>8..16+length*16</td>
<td>4</td>
<td>Float</td>
<td>Blue (0..1)</td>
</tr>
<tr>
<td>8..20+length*16</td>
<td>4</td>
<td>Float</td>
<td>Alpha (0..1)</td>
</tr>
</tbody>
</table>
5.1 GDScript

5.1.1 GDScript

Introduction

GDScript is a high level, dynamically typed programming language used to create content. It uses a syntax similar to Python (blocks are indent-based and many keywords are similar). Its goal is to be optimized for and tightly integrated with Godot Engine, allowing great flexibility for content creation and integration.

History

Initially, Godot was designed to support multiple scripting languages (this ability still exists today). However, only GDScript is in use right now. There is a little history behind this.

In the early days, the engine used the Lua scripting language. Lua is fast, but creating bindings to an object oriented system (by using fallbacks) was complex and slow and took an enormous amount of code. After some experiments with Python, it also proved difficult to embed.

The last third party scripting language that was used for shipped games was Squirrel, but it was dropped as well. At that point, it became evident that a custom scripting language could more optimally make use of Godot’s particular architecture:

- Godot embeds scripts in nodes. Most languages are not designed with this in mind.
- Godot uses several built-in data types for 2D and 3D math. Script languages do not provide this, and binding them is inefficient.
- Godot uses threads heavily for lifting and initializing data from the net or disk. Script interpreters for common languages are not friendly to this.
- Godot already has a memory management model for resources, most script languages provide their own, which results in duplicate effort and bugs.
- Binding code is always messy and results in several failure points, unexpected bugs and generally low maintainability.

The result of these considerations is GDScript. The language and interpreter for GDScript ended up being smaller than the binding code itself for Lua and Squirrel, while having equal functionality. With time, having a built-in language has proven to be a huge advantage.
Example of GDScript

Some people can learn better by just taking a look at the syntax, so here’s a simple example of how GDScript looks.

```gdscript
# a file is a class!
# inheritance
extends BaseClass
# member variables
var a = 5
var s = "Hello"
var arr = [1, 2, 3]
var dict = {"key":"value", 2:3}
# constants
const answer = 42
const thename = "Charly"
# enums
enum {UNIT_NEUTRAL, UNIT_ENEMY, UNIT_ALLY}
enum Named {THING_1, THING_2, ANOTHER_THING = -1}
# built-in vector types
var v2 = Vector2(1, 2)
var v3 = Vector3(1, 2, 3)
# function
func some_function(param1, param2):
    var local_var = 5
    if param1 < local_var:
        print(param1)
    elif param2 > 5:
        print(param2)
    else:
        print("fail!")
    for i in range(20):
        print(i)
    while(param2 != 0):
        param2 -= 1
    var local_var2 = param1+3
    return local_var2
# inner class
class Something:
    var a = 10
```
# constructor

```gdscript
func _init():
    print("constructed!")
    var lv = Something.new()
    print(lv.a)
```

If you have previous experience with statically typed languages such as C, C++, or C# but never used a dynamically typed one before, it is advised you read this tutorial: GDScript more efficiently.

## Language

In the following, an overview is given to GDScript. Details, such as which methods are available to arrays or other objects, should be looked up in the linked class descriptions.

### Identifiers

Any string that restricts itself to alphabetic characters (a to z and A to Z), digits (0 to 9) and _ qualifies as an identifier. Additionally, identifiers must not begin with a digit. Identifiers are case-sensitive (foo is different from FOO).

### Keywords

The following is the list of keywords supported by the language. Since keywords are reserved words (tokens), they can’t be used as identifiers.
<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>if</td>
<td>See <code>if/else/elif</code></td>
</tr>
<tr>
<td>elif</td>
<td>See <code>if/else/elif</code></td>
</tr>
<tr>
<td>else</td>
<td>See <code>if/else/elif</code></td>
</tr>
<tr>
<td>for</td>
<td>See <code>for</code></td>
</tr>
<tr>
<td>do</td>
<td>Reserved for future implementation of do...while loops.</td>
</tr>
<tr>
<td>while</td>
<td>See <code>while</code></td>
</tr>
<tr>
<td>switch</td>
<td>Reserved for future implementation.</td>
</tr>
<tr>
<td>case</td>
<td>Reserved for future implementation.</td>
</tr>
<tr>
<td>break</td>
<td>Exits the execution of the current <code>for</code> or <code>while</code> loop.</td>
</tr>
<tr>
<td>continue</td>
<td>Immediately skips to the next iteration of the <code>for</code> or <code>while</code> loop.</td>
</tr>
<tr>
<td>pass</td>
<td>Used where a statement is required syntactically but execution of code is undesired, e.g. in empty functions.</td>
</tr>
<tr>
<td>return</td>
<td>Returns a value from a function.</td>
</tr>
<tr>
<td>class</td>
<td>Defines a class.</td>
</tr>
<tr>
<td>extends</td>
<td>Defines what class to extend with the current class. Also tests whether a variable extends a given class.</td>
</tr>
<tr>
<td>tool</td>
<td>Executes the script in the editor.</td>
</tr>
<tr>
<td>signal</td>
<td>Defines a signal.</td>
</tr>
<tr>
<td>func</td>
<td>Defines a function.</td>
</tr>
<tr>
<td>static</td>
<td>Defines a static function. Static member variables are not allowed.</td>
</tr>
<tr>
<td>const</td>
<td>Defines a constant.</td>
</tr>
<tr>
<td>enum</td>
<td>Defines an enum.</td>
</tr>
<tr>
<td>var</td>
<td>Defines a variable.</td>
</tr>
<tr>
<td>onready</td>
<td>Initializes a variable once the Node the script is attached to and its children are part of the scene tree.</td>
</tr>
<tr>
<td>export</td>
<td>Saves a variable along with the resource it’s attached to and makes it visible and modifiable in the editor.</td>
</tr>
<tr>
<td>setget</td>
<td>Defines setter and getter functions for a variable.</td>
</tr>
<tr>
<td>breakpoint</td>
<td>Editor helper for debugger breakpoints.</td>
</tr>
</tbody>
</table>

**Operators**

The following is the list of supported operators and their precedence (TODO, change since this was made to reflect python operators)
### Operator

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x[index]</td>
<td>Subscription, Highest Priority</td>
</tr>
<tr>
<td>x.attribute</td>
<td>Attribute Reference</td>
</tr>
<tr>
<td>extends</td>
<td>Instance Type Checker</td>
</tr>
<tr>
<td>~</td>
<td>Bitwise NOT</td>
</tr>
<tr>
<td>-x</td>
<td>Negative</td>
</tr>
<tr>
<td>* / %</td>
<td>Multiplication / Division / Remainder</td>
</tr>
<tr>
<td>+ -</td>
<td>Addition / Subtraction</td>
</tr>
<tr>
<td>&lt; &gt; &gt;&gt;</td>
<td>Bit Shifting</td>
</tr>
<tr>
<td>&amp;</td>
<td>Bitwise AND</td>
</tr>
<tr>
<td>^</td>
<td>Bitwise XOR</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;&gt; == != =&gt; &lt;=</td>
<td>Comparisons</td>
</tr>
<tr>
<td>in</td>
<td>Content Test</td>
</tr>
<tr>
<td>! not</td>
<td>Boolean NOT</td>
</tr>
<tr>
<td>and &amp;&amp;</td>
<td>Boolean AND</td>
</tr>
<tr>
<td>or</td>
<td></td>
</tr>
<tr>
<td>if x else</td>
<td>Ternary if/else</td>
</tr>
<tr>
<td>= += -= *= /= %= &amp;=</td>
<td>=</td>
</tr>
</tbody>
</table>

#### Literals

<table>
<thead>
<tr>
<th>Literal</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>45</td>
<td>Base 10 integer</td>
</tr>
<tr>
<td>0x8F51</td>
<td>Base 16 (hex) integer</td>
</tr>
<tr>
<td>3.14, 58.1e-10</td>
<td>Floating point number (real)</td>
</tr>
<tr>
<td>&quot;Hello&quot;, &quot;Hi&quot;</td>
<td>Strings</td>
</tr>
<tr>
<td>&quot;<em><strong>Hello,Dude</strong></em>&quot;</td>
<td>Multiline string</td>
</tr>
<tr>
<td>@&quot;Node/Label&quot;</td>
<td>NodePath or StringName</td>
</tr>
</tbody>
</table>

### Comments

Anything from a # to the end of the line is ignored and is considered a comment.

```godot
# This is a comment
```

### Built-in types

#### Basic built-in types

A variable in GDScript can be assigned to several built-in types.
null

null is an empty data type that contains no information and can not be assigned any other value.

bool

The Boolean data type can only contain true or false.

int

The integer data type can only contain integer numbers, (both negative and positive).

float

Used to contain a floating point value (real numbers).

String

A sequence of characters in Unicode format. Strings can contain the standard C escape sequences. GDScript supports format strings aka printf functionality.

Vector built-in types

Vector2

2D vector type containing x and y fields. Can alternatively access fields as width and height for readability. Can also be accessed as array.

Rect2

2D Rectangle type containing two vectors fields: pos and size. Alternatively contains an end field which is pos+size.

Vector3

3D vector type containing x, y and z fields. This can also be accessed as an array.

Matrix32

3x2 matrix used for 2D transforms.

Plane

3D Plane type in normalized form that contains a normal vector field and a d scalar distance.
Quat

Quaternion is a datatype used for representing a 3D rotation. It’s useful for interpolating rotations.

AABB

Axis Aligned bounding box (or 3D box) contains 2 vectors fields: \text{pos} and \text{size}. Alternatively contains an \text{end} field which is \text{pos}+\text{size}. As an alias of this type, \text{Rect3} can be used interchangeably.

Matrix3

3x3 matrix used for 3D rotation and scale. It contains 3 vector fields (x, y and z) and can also be accessed as an array of 3D vectors.

Transform

3D Transform contains a Matrix3 field \text{basis} and a Vector3 field \text{origin}.

Engine built-in types

Color

Color data type contains r, g, b, and a fields. It can also be accessed as h, s, and v for hue/saturation/value.

Image

Contains a custom format 2D image and allows direct access to the pixels.

NodePath

Compiled path to a node used mainly in the scene system. It can be easily assigned to, and from, a String.

RID

Resource ID (RID). Servers use generic RIDs to reference opaque data.

Object

Base class for anything that is not a built-in type.

InputEvent

Events from input devices are contained in very compact form in InputEvent objects. Due to the fact that they can be received in high amounts from frame to frame they are optimized as their own data type.
Container built-in types

Array

Generic sequence of arbitrary object types, including other arrays or dictionaries (see below). The array can resize dynamically. Arrays are indexed starting from index 0. Starting with Godot 2.1, indices may be negative like in Python, to count from the end.

```gdscript
var arr=[]
arr=[1, 2, 3]
var b = arr[1]  # this is 2
var c = arr[arr.size()-1]  # this is 3
var d = arr[-1]  # same as the previous line, but shorter
arr[0] = "Hi!"
# replacing value 1 with "Hi"
arr.append(4)
# array is now ["Hi", 2, 3, 4]
```

GDScript arrays are allocated linearly in memory for speed. Very large arrays (more than tens of thousands of elements) may however cause memory fragmentation. If this is a concern special types of arrays are available. These only accept a single data type. They avoid memory fragmentation and also use less memory but are atomic and tend to run slower than generic arrays. They are therefore only recommended to use for very large data sets:

- **ByteArray**: An array of bytes (integers from 0 to 255).
- **IntArray**: An array of integers.
- **FloatArray**: An array of floats.
- **StringArray**: An array of strings.
- **Vector2Array**: An array of `Vector2` objects.
- **Vector3Array**: An array of `Vector3` objects.
- **ColorArray**: An array of `Color` objects.

Dictionary

Associative container which contains values referenced by unique keys.

```gdscript
var d={4:5, "a key":"a value", 28:[1,2,3]}
d["Hi!"] = 0
var d = {
    22 : "Value",
    "somekey" : 2,
    "otherkey" : [2,3,4],
    "morekey" : "Hello"
}
```

Lua-style table syntax is also supported. Lua-style uses `=` instead of `:` and doesn’t use quotes to mark string keys (making for slightly less to write). Note however that like any GDScript identifier, keys written in this form cannot start with a digit.

```gdscript
var d = {
    test22 = "Value",
    somekey = 2,
    otherkey = [2,3,4],
    morekey = "Hello"
}
```
To add a key to an existing dictionary, access it like an existing key and assign to it:

```godot
def = {} # create an empty Dictionary
def.Waiting = 14 # add String "Waiting" as a key and assign the value 14 to it
def[4] = "hello" # add integer 4 as a key and assign the String "hello" as its value
def["Godot"] = 3.01 # add String "Godot" as a key and assign the value 3.01 to it
```

### Data

#### Variables

Variables can exist as class members or local to functions. They are created with the `var` keyword and may, optionally, be assigned a value upon initialization.

```godot
def a # data type is null by default
def b = 5
def c = 3.8
def d = b + c # variables are always initialized in order
```

#### Constants

Constants are similar to variables, but must be constants or constant expressions and must be assigned on initialization.

```godot
def const a = 5
def const b = Vector2(20, 20)
def const c = 10 + 20 # constant expression
def const d = Vector2(20, 30).x # constant expression: 20
def const e = [1, 2, 3, 4][0] # constant expression: 1
def const f = sin(20) # sin() can be used in constant expressions
def const g = x + 20 # invalid; this is not a constant expression!
```

### Enums

 Enums are basically a shorthand for constants, and are pretty useful if you want to assign consecutive integers to some constant.

If you pass a name to the enum, it would also put all the values inside a constant dictionary of that name.

```godot
def enum {TILE_BRICK, TILE_FLOOR, TILE_SPIKE, TILE_TELEPORT}
# is the same as:
def const TILE_BRICK = 0
def const TILE_FLOOR = 1
def const TILE_SPIKE = 2
def const TILE_TELEPORT = 3

def enum State {STATE_IDLE, STATE_JUMP = 5, STATE_SHOOT}
# is the same as:
def const STATE_IDLE = 0
def const STATE_JUMP = 5
def const STATE_SHOOT = 6
def const State = {STATE_IDLE = 0, STATE_JUMP = 5, STATE_SHOOT = 6}
```
Functions

Functions always belong to a class. The scope priority for variable look-up is: local → class member → global. The self variable is always available and is provided as an option for accessing class members, but is not always required (and should not be sent as the function’s first argument, unlike Python).

```godot
func myfunction(a, b):
    print(a)
    print(b)
    return a + b  # return is optional; without it null is returned
```

A function can return at any point. The default return value is null.

Referencing Functions

To call a function in a base class (i.e. one extend-ed in your current class), prepend . to the function name:

```godot
.mybasefunc(args)
```

Contrary to Python, functions are not first class objects in GDScript. This means they cannot be stored in variables, passed as an argument to another function or be returned from other functions. This is for performance reasons.

To reference a function by name at runtime, (e.g. to store it in a variable, or pass it to another function as an argument) one must use the call or funcref helpers:

```godot
# Call a function by name in one step
mynode.call("myfunction", args)

# Store a function reference
var myfunc = funcref(mynode, "myfunction")
# Call stored function reference
myfunc.call_func(args)
```

Remember that default functions like _init, and most notifications such as _enter_tree, _exit_tree, _process, _fixed_process, etc. are called in all base classes automatically. So there is only a need to call the function explicitly when overloading them in some way.

Static functions

A function can be declared static. When a function is static it has no access to the instance member variables or self. This is mainly useful to make libraries of helper functions:

```godot
static func sum2(a, b):
    return a + b
```

Statements and control flow

Statements are standard and can be assignments, function calls, control flow structures, etc (see below). ; as a statement separator is entirely optional.
if/else/elif

Simple conditions are created by using the if/else/elif syntax. Parenthesis around conditions are allowed, but not required. Given the nature of the tab-based indentation, elif can be used instead of else/if to maintain a level of indentation.

```python
if [expression]:
    statement(s)
elif [expression]:
    statement(s)
else:
    statement(s)
```

Short statements can be written on the same line as the condition:

```python
if (1 + 1 == 2): return 2 + 2
else:
    var x = 3 + 3
    return x
```

Sometimes you might want to assign a different initial value based on a boolean expression. In this case ternary-if expressions come in handy:

```python
var x = [true-value] if [expression] else [false-value]
y += 3 if y < 10 else -1
```

while

Simple loops are created by using while syntax. Loops can be broken using break or continued using continue:

```python
while [expression]:
    statement(s)
```

for

To iterate through a range, such as an array or table, a for loop is used. When iterating over an array, the current array element is stored in the loop variable. When iterating over a dictionary, the index is stored in the loop variable.

```python
for x in [5, 7, 11]:
    statement # loop iterates 3 times with x as 5, then 7 and finally 11

var dict = {"a":0, "b":1, "c":2}
for i in dict:
    print(dict[i]) # loop provides the keys in an arbitrary order; may print 0, 1, 2,
    → or 2, 0, 1, etc...

for i in range(3):
    statement # similar to [0, 1, 2] but does not allocate an array

for i in range(1,3):
    statement # similar to [1, 2] but does not allocate an array

for i in range(2,8,2):
    statement # similar to [2, 4, 6] but does not allocate an array
```
for c in "Hello":
    print(c)  # iterate through all characters in a String, print every letter on new line

Classes

By default, the body of a script file is an unnamed class and it can only be referenced externally as a resource or file. Class syntax is meant to be very compact and can only contain member variables or functions. Static functions are allowed, but not static members (this is in the spirit of thread safety, since scripts can be initialized in separate threads without the user knowing). In the same way, member variables (including arrays and dictionaries) are initialized every time an instance is created.

Below is an example of a class file.

# saved as a file named myclass.gd
var a = 5
func print_value_of_a():
    print(a)

Inheritance

A class (stored as a file) can inherit from

• A global class
• Another class file
• An inner class inside another class file.

Multiple inheritance is not allowed.

Inheritance uses the `extends` keyword:

# Inherit/extend a globally available class
extends SomeClass

# Inherit/extend a named class file
extends "somefile.gd"

# Inherit/extend an inner class in another file
extends "somefile.gd".SomeInnerClass

To check if a given instance inherits from a given class the `extends` keyword can be used as an operator instead:

# Cache the enemy class
const enemy_class = preload("enemy.gd")

# [...]  

# use 'extends' to check inheritance
if (entity extends enemy_class):
    entity.apply_damage()
Class Constructor

The class constructor, called on class instantiation, is named _init. As mentioned earlier, the constructors of parent classes are called automatically when inheriting a class. So there is usually no need to call .__init__() explicitly.

If a parent constructor takes arguments, they are passed like this:

```gdscript
func __init__(args)(parent_args):
pass
```

Inner classes

A class file can contain inner classes. Inner classes are defined using the class keyword. They are instanced using the ClassName.new() function.

```gdscript
# inside a class file
# An inner class in this class file
class SomeInnerClass:
    var a = 5
    func print_value_of_a():
        print(a)

# This is the constructor of the class file's main class
func __init__():
    var c = SomeInnerClass.new()
    c.print_value_of_a()
```

Classes as resources

Classes stored as files are treated as resources. They must be loaded from disk to access them in other classes. This is done using either the load or preload functions (see below). Instancing of a loaded class resource is done by calling the new function on the class object:

```gdscript
# Load the class resource when calling load()
var MyClass = load("myclass.gd")

# Preload the class only once at compile time
var MyClass2 = preload("myclass.gd")

func __init__():
    var a = MyClass.new()
    a.somefunction()
```

Exports

Class members can be exported. This means their value gets saved along with the resource (e.g. the scene) they’re attached to. They will also be available for editing in the property editor. Exporting is done by using the export keyword:

```gdscript
extends Button

export var number = 5  # value will be saved and visible in the property editor
```
An exported variable must be initialized to a constant expression or have an export hint in the form of an argument to the export keyword (see below).

One of the fundamental benefits of exporting member variables is to have them visible and editable in the editor. This way artists and game designers can modify values that later influence how the program runs. For this, a special export syntax is provided.

```godot
# If the exported value assigns a constant or constant expression,
# the type will be inferred and used in the editor
export var number = 5

# Export can take a basic data type as an argument which will be
# used in the editor
export (int) var number

# Export can also take a resource type to use as a hint
export (Texture) var character_face
export (PackedScene) var scene_file

# Integers and strings hint enumerated values
# Editor will enumerate as 0, 1 and 2
export (int, "Warrior", "Magician", "Thief") var character_class
# Editor will enumerate with string names
export (String, "Rebecca", "Mary", "Leah") var character_name

# Strings as paths
# String is a path to a file
export (String, FILE) var f
# String is a path to a directory
export (String, DIR) var f
# String is a path to a file, custom filter provided as hint
export (String, FILE, "*.txt") var f

# Using paths in the global filesystem is also possible,
# but only in tool scripts (see further below)

# String is a path to a PNG file in the global filesystem
export (String, FILE, GLOBAL, "*.png") var tool_image
# String is a path to a directory in the global filesystem
export (String, DIR, GLOBAL) var tool_dir

# The MULTILINE setting tells the editor to show a large input
# field for editing over multiple lines
export (String, MULTILINE) var text

# Limiting editor input ranges

# Allow integer values from 0 to 20
export (int, 20) var i
# Allow integer values from -10 to 20
export (int, -10, 20) var j
# Allow floats from -10 to 20, with a step of 0.2
```
It must be noted that even if the script is not being run while at the editor, the exported properties are still editable (see below for “tool”).

### Exporting bit flags

Integers used as bit flags can store multiple true/false (boolean) values in one property. By using the export hint `int,FLAGS`, they can be set from the editor:

```gdscript
# Individually edit the bits of an integer
export(int, FLAGS) var spell_elements = ELEMENT_WIND | ELEMENT_WATER
```

Restricting the flags to a certain number of named flags is also possible. The syntax is very similar to the enumeration syntax:

```gdscript
# Set any of the given flags from the editor
export(int, FLAGS, "Fire", "Water", "Earth", "Wind") var spell_elements = 0
```

In this example, Fire has value 1, Water has value 2, Earth has value 4 and Wind corresponds to value 8. Usually, constants should be defined accordingly (e.g. `const ELEMENT_WIND = 8` and so on).

Using bit flags requires some understanding of bitwise operations. If in doubt, boolean variables should be exported instead.

### Exporting arrays

Exporting arrays works but with an important caveat: While regular arrays are created local to every class instance, exported arrays are shared between all instances. This means that editing them in one instance will cause them to change in all other instances. Exported arrays can have initializers, but they must be constant expressions.

```gdscript
# Exported array, shared between all instances.
# Default value must be a constant expression.
```
export var a=[1,2,3]

# Typed arrays also work, only initialized empty:

export var vector3s = Vector3Array()
export var strings = StringArray()

# Regular array, created local for every instance.
# Default value can include run-time values, but can't
# be exported.

var b = [a,2,3]

Setters/getters

It is often useful to know when a class’ member variable changes for whatever reason. It may also be desired to
encapsulate its access in some way.

For this, GDScript provides a setter/getter syntax using the setget keyword. It is used directly after a variable
definition:

```gdscript
var variable = value setget setterfunc, getterfunc
```

Whenever the value of `variable` is modified by an external source (i.e. not from local usage in the class), the `setter`
function (`setterfunc` above) will be called. This happens before the value is changed. The `setter` must decide what
to do with the new value. Vice-versa, when `variable` is accessed, the `getter` function (`getterfunc` above) must
return the desired value. Below is an example:

```gdscript
var myvar setget myvar_set, myvar_get

func myvar_set(newvalue):
    myvar = newvalue

func myvar_get():
    return myvar # getter must return a value
```

Either of the `setter` or `getter` functions can be omitted:

```gdscript
# Only a setter
var myvar = 5 setget myvar_set
# Only a getter (note the comma)
var myvar = 5 setget ,myvar_get
```

Get/Setters are especially useful when exporting variables to editor in tool scripts or plugins, for validating input.

As said local access will not trigger the setter and getter. Here is an illustration of this:

```gdscript
func _init():
    # Does not trigger setter/getter
    myinteger=5
    print(myinteger)

    # Does trigger setter/getter
    self.myinteger=5
    print(self.myinteger)
```
Tool mode

Scripts, by default, don’t run inside the editor and only the exported properties can be changed. In some cases it is desired that they do run inside the editor (as long as they don’t execute game code or manually avoid doing so). For this, the `tool` keyword exists and must be placed at the top of the file:

```
tool
extends Button

func _ready():
    print("Hello")
```

Memory management

If a class inherits from `Reference`, then instances will be freed when no longer in use. No garbage collector exists, just simple reference counting. By default, all classes that don’t define inheritance extend `Reference`. If this is not desired, then a class must inherit `Object` manually and must call `instance.free()`. To avoid reference cycles that can’t be freed, a `weakref` function is provided for creating weak references.

Signals

It is often desired to send a notification that something happened in an instance. GDScript supports creation of built-in Godot signals. Declaring a signal in GDScript is easy using the `signal` keyword.

```
# No arguments
signal your_signal_name

# With arguments
signal your_signal_name_with_args(a,b)
```

These signals, just like regular signals, can be connected in the editor or from code. Just take the instance of a class where the signal was declared and connect it to the method of another instance:

```
func _callback_no_args():
    print("Got callback!")

func _callback_args(a,b):
    print("Got callback with args! a: ",a," and b: ",b)

func _at_some_func():
    instance.connect("your_signal_name",self,"_callback_no_args")
    instance.connect("your_signal_name_with_args",self,"_callback_args")
```

It is also possible to bind arguments to a signal that lacks them with your custom values:

```
func _at_some_func():
    instance.connect("your_signal_name",self,"_callback_args",[22,"hello"])
```

This is very useful when a signal from many objects is connected to a single callback and the sender must be identified:

```
func _button_pressed(which):
    print("Button was pressed: ",which.get_name())

func _ready():
```
for b in get_node("buttons").get_children():
    b.connect("pressed", self,"_button_pressed",[b])

Finally, emitting a custom signal is done by using the Object.emit_signal method:

```gdscript
func _at_some_func():
    emit_signal("your_signal_name")
    emit_signal("your_signal_name_with_args",55,128)
    someinstance.emit_signal("some_signal")
```

## Coroutines

GDScript offers support for coroutines via the `yield` built-in function. Calling `yield()` will immediately return from the current function, with the current frozen state of the same function as the return value. Calling `resume` on this resulting object will continue execution and return whatever the function returns. Once resumed the state object becomes invalid. Here is an example:

```gdscript
func myfunc():
    print("hello")
    yield()
    print("world")

func _ready():
    var y = myfunc()
    # Function state saved in 'y'
    print("my dear")
    y.resume()
    # 'y' resumed and is now an invalid state
```

Will print:

```
hello
my dear
world
```

It is also possible to pass values between `yield()` and `resume()`, for example:

```gdscript
func myfunc():
    print("hello")
    print( yield() )
    return "cheers!"

func _ready():
    var y = myfunc()
    # Function state saved in 'y'
    print( y.resume("world") )
    # 'y' resumed and is now an invalid state
```

Will print:

```
334 Chapter 5. Scripting
```
Coroutines & signals

The real strength of using \texttt{yield} is when combined with signals. \texttt{yield} can accept two parameters, an object and a signal. When the signal is received, execution will recommence. Here are some examples:

```gdscript
# Resume execution the next frame
yield( get_tree(), "idle_frame" )

# Resume execution when animation is done playing:
yield( get_node("AnimationPlayer"), "finished" )
```

Onready keyword

When using nodes, it’s very common to desire to keep references to parts of the scene in a variable. As scenes are only warranted to be configured when entering the active scene tree, the sub-nodes can only be obtained when a call to \texttt{Node._ready()} is made.

```gdscript
var mylabel
func _ready():
    mylabel = get_node("MyLabel")
```

This can get a little cumbersome, specially when nodes and external references pile up. For this, GDScript has the \texttt{onready} keyword, that defers initialization of a member variable until \texttt{_ready} is called. It can replace the above code with a single line:

```gdscript
onready var mylabel = get_node("MyLabel")
```

5.1.2 GDScript more efficiently

About

This tutorial aims to be a quick reference for how to use GDScript more efficiently. It focuses on common cases specific to the language, but also covers a lot of information on dynamically typed languages.

It’s meant to be specially useful for programmers with little or no previous experience with dynamically typed languages.

Dynamic nature

Pros & cons of dynamic typing

GDScript is a Dynamically Typed language. As such, it’s main advantages are that:

- The language is very simple to learn.
- Most code can be written and changed quickly and without hassle.
• Less code written means less errors & mistakes to fix.
• Easier to read the code (less clutter).
• No compilation is required to test.
• Runtime is tiny.
• Duck-typing and polymorphism by nature.

While the main disadvantages are:
• Less performance than statically typed languages.
• More difficult to refactor (symbols can’t be traced)
• Some errors that would typically be detected at compile time in statically typed languages only appear while running the code (because expression parsing is more strict).
• Less flexibility for code-completion (some variable types are only known at run-time).

This, translated to reality, means that Godot+GDScript are a combination designed to create games very quickly and efficiently. For games that are very computationally intensive and can’t benefit from the engine built-in tools (such as the Vector types, Physics Engine, Math library, etc), the possibility of using C++ is present too. This allows to still create the entire game in GDScript and add small bits of C++ in the areas that need a performance boost.

**Variables & assignment**

All variables in a dynamically typed language are “variant”-like. This means that their type is not fixed, and is only modified through assignment. Example:

Static:

```c
int a; // value uninitialized
a = 5;  // this is valid
a = "Hi!";  // this is invalid
```

Dynamic:

```c
var a  # null by default
a = 5  # valid, 'a' becomes an integer
a = "Hi!"  # valid, 'a' changed to a string
```

**As function arguments:**

Functions are of dynamic nature too, which means they can be called with different arguments, for example:

Static:

```c
void print_value(int value)
{
    printf("value is %i\n",value);
}

[...]

print_value(55);  // valid
print_value("Hello");  // invalid
```
Dynamic:

```godot
func print_value(value):
    print(value)
[..]

print_value(55)  # valid
print_value("Hello")  # valid
```

Pointers & referencing:

In static languages such as C or C++ (and to some extent Java and C#), there is a distinction between a variable and a pointer/reference to a variable. The latter allows the object to be modified by other functions by passing a reference to the original one.

In C# or Java, everything not a built-in type (int, float, sometimes String) is always a pointer or a reference. References are also garbage-collected automatically, which means they are erased when no longer used. Dynamically typed languages tend to use this memory model too. Some Examples:

  • C++:

    ```cpp
    void use_class(SomeClass *instance) {
        instance->use();
    }
    void do_something() {
        SomeClass *instance = new SomeClass; // created as pointer
        use_class(instance); // passed as pointer
        delete instance; // otherwise it will leak memory
    }
    ```

  • Java:

    ```java
    @Override
    public final void use_class(SomeClass instance) {
        instance.use();
    }
    public final void do_something() {
        SomeClass instance = new SomeClass(); // created as reference
        use_class(instance); // passed as reference
        // garbage collector will get rid of it when not in
        // use and freeze your game randomly for a second
    }
    ```

  • GDScript:

    ```gdscript
    func use_class(instance);  # does not care about class type
        instance.use()  # will work with any class that has a ".use()" method.
    func do_something();
        var instance = SomeClass.new()  # created as reference
    ```
In GDScript, only base types (int, float, string and the vector types) are passed by value to functions (value is copied). Everything else (instances, arrays, dictionaries, etc) is passed as reference. Classes that inherit `Reference` (the default if nothing is specified) will be freed when not used, but manual memory management is allowed too if inheriting manually from `Object`.

### Arrays

Arrays in dynamically typed languages can contain many different mixed datatypes inside and are always dynamic (can be resized at any time). Compare for example arrays in statically typed languages:

```cpp
int *array = new int[4]; // create array
array[0] = 10; // initialize manually
array[1] = 20; // can't mix types
array[2] = 40;
array[3] = 60; // can't resize
use_array(array); // passed as pointer
delete[] array; // must be freed
// or
std::vector<int> array;
array.resize(4);
array[0] = 10; // initialize manually
array[1] = 20; // can't mix types
array[2] = 40;
array[3] = 60;
array.resize(3); // can be resized
use_array(array); // passed reference or value
// freed when stack ends
```

And in GDScript:

```gd
var array = [10, "hello", 40, 60] # simple, and can mix types
array.resize(3) # can be resized
use_array(array) # passed as reference
# freed when no longer in use
```

In dynamically typed languages, arrays can also double as other datatypes, such as lists:

```gd
var array = []
array.append(4)
array.append(5)
array.pop_front()
```

Or unordered sets:

```gd
var a = 20
if a in [10, 20, 30]:
    print("We have a winner!")
```
Dictionaries

Dictionaries are always a very powerful in dynamically typed languages. Most programmers that come from statically typed languages (such as C++ or C#) ignore their existence and make their life unnecessarily more difficult. This datatype is generally not present in such languages (or only on limited form).

Dictionaries can map any value to any other value with complete disregard for the datatype used as either key or value. Contrary to popular belief, they are very efficient because they can be implemented with hash tables. They are, in fact, so efficient that some languages will go as far as implementing arrays as dictionaries.

Example of Dictionary:

```gdscript
var d = { "name": "john", "age": 22 } # simple syntax
print("Name: ", d["name"], " Age: ", d["age"])
```

Dictionaries are also dynamic, keys can be added or removed at any point at little cost:

```
d["mother"] = "Rebecca" # addition
d["age"] = 11 # modification
d.erase("name") # removal
```

In most cases, two-dimensional arrays can often be implemented more easily with dictionaries. Here’s a simple battleship game example:

```
# battleship game

const SHIP = 0
const SHIP_HIT = 1
const WATER_HIT = 2

var board = {}

func initialize():
    board[Vector(1,1)] = SHIP
    board[Vector(1,2)] = SHIP
    board[Vector(1,3)] = SHIP

func missile(pos):
    if pos in board: # something at that pos
        if board[pos] == SHIP: # there was a ship! hit it
            board[pos] = SHIP_HIT
        else:
            print("already hit here!") # hey dude you already hit here
    else: # nothing, mark as water
        board[pos] = WATER_HIT

func game():
    initialize()
    missile(Vector2(1,1))
    missile(Vector2(5,8))
    missile(Vector2(2,3))
```

Dictionaries can also be used as data markup or quick structures. While GDScript dictionaries resemble python dictionaries, it also supports Lua style syntax and indexing, which makes it very useful for writing initial states and quick structs.
# same example, lua-style support
# this syntax is a lot more readable and usable

```lua
var d = {
    name = "john",
    age = 22
}

print("Name: ", d.name, " Age: ", d.age) # used "." based indexing

# indexing

d["mother"] = "rebecca" # this works

# this would work too to create a new key

d.mother = "caroline"
```

## For & while

Iterating in some statically typed languages can be quite complex:

```c
const char* strings = new const char*[50];

[[]]

for(int i=0; i<50; i++)
{
    printf("value: %s\n", i, strings[i]);
}

// even in STL:

for(std::list<std::string>::const_iterator it = strings.begin(); it != strings.end(); it++) {
    std::cout << *it << std::endl;
}
```

This is usually greatly simplified in dynamically typed languages:

```python
for s in strings:
    print(s)
```

Container datatypes (arrays and dictionaries) are iterable. Dictionaries allow iterating the keys:

```python
for key in dict:
    print(key, " -> ", dict[key])
```

Iterating with indices is also possible:

```python
for i in range(strings.size()):
    print(strings[i])
```

The range() function can take 3 arguments:

- `range(n)` (will go from 0 to n-1)
- `range(b, n)` (will go from b to n-1)
- `range(b, n, s)` (will go from b to n-1, in steps of s)
Some examples:

```plaintext
for(int i=0; i<10; i++) {}  
for(int i=5; i<10; i++) {}  
for(int i=5; i<10; i+=2) {}
```

Translate to:

```plaintext
for i in range(10):
    for i in range(5, 10):
        for i in range(5, 10, 2):
```

And backwards looping is done through a negative counter:

```plaintext
for(int i=10; i>0; i--) {}  
```

becomes

```plaintext
for i in range(10, 0, -1):
```

**While**

while() loops are the same everywhere:

```plaintext
var i = 0
while(i < strings.size()):
    print(strings[i])
    i += 1
```

**Duck typing**

One of the most difficult concepts to grasp when moving from a statically typed language to a dynamic one is duck typing. Duck typing makes overall code design much simpler and straightforward to write, but it’s not obvious how it works.

As an example, imagine a situation where a big rock is falling down a tunnel, smashing everything on its way. The code for the rock, in a statically typed language would be something like:

```plaintext
void BigRollingRock::on_object_hit(Smashable *entity)
{
    entity->smash();
}
```

This, way, everything that can be smashed by a rock would have to inherit Smashable. If a character, enemy, piece of furniture, small rock were all smashable, they would need to inherit from the class Smashable, possibly requiring multiple inheritance. If multiple inheritance was undesired, then they would have to inherit a common class like Entity. Yet, it would not be very elegant to add a virtual method `smash()` to Entity only if a few of them can be smashed.

5.1. GDScrip
With dynamically typed languages, this is not a problem. Duck typing makes sure you only have to define a `smash()` function where required and that’s it. No need to consider inheritance, base classes, etc.

```python
func _on_object_hit(object):
    object.smash()
```

And that’s it. If the object that hit the big rock has a `smash()` method, it will be called. No need for inheritance or polymorphism. Dynamically typed languages only care about the instance having the desired method or member, not what it inherits or the class type. The definition of Duck Typing should make this clearer:

“When I see a bird that walks like a duck and swims like a duck and quacks like a duck, I call that bird a duck”

In this case, it translates to:

“If the object can be smashed, don’t care what it is, just smash it.”

Yes, we should call it Hulk typing instead. Anyway though, there exists the possibility of the object being hit not having a `smash()` function. Some dynamically typed languages simply ignore a method call when it doesn’t exist (like Objective C), but GDScript is more strict, so checking if the function exists is desirable:

```python
func _on_object_hit(object):
    if (object.has_method("smash")):
        object.smash()
```

Then, simply define that method and anything the rock touches can be smashed.

### 5.1.3 GDScript format strings

GDScript offers a feature called `format strings` which allows reusing text templates to succinctly create different but similar strings.

Format strings are just like normal strings, except they contain certain placeholder character-sequences. These placeholders can then easily be replaced by parameters handed to the format string.

As an example, with `%s` as a placeholder, the format string "Hello %s, how are you?" can easily be changed to "Hello World, how are you?". Notice the placeholder is in the middle of the string; modifying it without format strings could be cumbersome.

#### Usage in GDScript

Examine this concrete GDScript example:

```python
# Define a format string with placeholder '%s'
var format_string = "We're waiting for %s."

# Using the '%' operator, the placeholder is replaced with the desired value
var actual_string = format_string % "Godot"

print(actual_string)
# output: "We're waiting for Godot."
```

Placeholders always start with a `%`, but the next character or characters, the `format specifier`, determines how the given value is converted to a string.

The `%s` seen in the example above is the simplest placeholder and works for most use cases: it converts the value by the same method by which an implicit String conversion or `str()` would convert it. Strings remain unchanged, Booleans turn into either "True" or "False", an integral or real number becomes a decimal, other types usually return their data in a human-readable string.
There are other format specifiers.

**Multiple placeholders**

Format strings may contain multiple placeholders. In such a case, the values are handed in the form of an array, one value per placeholder (unless using a format specifier with *, see *dynamic padding*):

```python
var format_string = "%s was reluctant to learn %s, but now he enjoys it."
var actual_string = format_string % ["Estragon", "GDScript"]

print(actual_string)
# output: "Estragon was reluctant to learn GDScript, but now he enjoys it."
```

Note the values are inserted in order. Remember all placeholders must be replaced at once, so there must be an appropriate number of values.

**Format specifiers**

There are format specifiers other than `s` that can be used in placeholders. They consist of one or more characters. Some of them work by themselves like `s`, some appear before other characters, some only work with certain values or characters.

**Placeholder types**

One and only one of these must always appear as the last character in a format specifier. Apart from `s`, these require certain types of parameters.

<table>
<thead>
<tr>
<th>Character</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>s</td>
<td>Simple</td>
<td>Conversion to String by the same method as implicit String conversion.</td>
</tr>
<tr>
<td>c</td>
<td>Single Unicode character</td>
<td>Expects an unsigned 8-bit integer (0-255) for a code point or a single-character string.</td>
</tr>
<tr>
<td>d</td>
<td>Decimal integral number</td>
<td>Expects an integral or real number (will be floored).</td>
</tr>
<tr>
<td>o</td>
<td>Octal integral number</td>
<td>Expects an integral or real number (will be floored).</td>
</tr>
<tr>
<td>x</td>
<td>Hexadecimal integral number with lower-case letters</td>
<td>Expects an integral or real number (will be floored).</td>
</tr>
<tr>
<td>X</td>
<td>Hexadecimal integral number with upper-case letters</td>
<td>Expects an integral or real number (will be floored).</td>
</tr>
<tr>
<td>f</td>
<td>Decimal real number</td>
<td>Expects an integral or real number.</td>
</tr>
</tbody>
</table>

**Placeholder modifiers**

These characters appear before the above. Some of them work only under certain conditions.

<table>
<thead>
<tr>
<th>Character</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>In number specifiers, <strong>show + sign</strong> if positive.</td>
</tr>
<tr>
<td>-</td>
<td>Pad to the right rather than the left.</td>
</tr>
<tr>
<td>.</td>
<td>Before <code>f</code>, set <strong>precision</strong> to 0 decimal places. Can be followed up with numbers to change. Padded with zeroes.</td>
</tr>
<tr>
<td>*</td>
<td><strong>Dynamic padding</strong>, expect additional integral parameter to set padding or precision after <code>.</code>, see <em>dynamic padding</em>.</td>
</tr>
</tbody>
</table>

5.1. GDScript
Padding

The . (dot), * (asterisk), - (minus sign) and digit (0-9) characters are used for padding. This allows printing several values aligned vertically as if in a column, provided a fixed-width font is used.

To pad a string to a minimum length, add an integer to the specifier:

```python
print("%10d" % 12345)
# output: " 12345"
# 5 leading spaces for a total length of 10
```

If the integer starts with 0, integral values are padded with zeroes instead of white space:

```python
print("%010d" % 12345)
# output: "0000012345"
```

Precision can be specified for real numbers by adding a . (dot) with an integer following it. With no integer after ., a precision of 0 is used, rounding to integral value. The integer to use for padding must appear before the dot.

```python
# pad to minimum length of 10, round to 3 decimal places
print("%10.3f" % 10000.5555)
# output: "10000.556"
# 1 leading space
```

The - character will cause padding to the right rather than the left, useful for right text alignment:

```python
print("%-10d" % 12345678)
# output: "12345678 
# 2 trailing spaces
```

Dynamic padding

By using the * (asterisk) character, the padding or precision can be set without modifying the format string. It is used in place of an integer in the format specifier. The values for padding and precision are then passed when formatting:

```python
var format_string = "%*.*f"
# pad to length of 7, round to 3 decimal places:
print(format_string % [7, 3, 8.8888])
# output: "8.889"
# 2 leading spaces
```

It is still possible to pad with zeroes in integer placeholders by adding 0 before *:

```python
print("%0*d" % [2, 3])
#output: "03"
```

Escape sequence

To insert a literal % character into a format string, it must be escaped to avoid reading it as a placeholder. This is done by doubling the character:

```python
var health = 56
print("Remaining health: %d%%" % health)
# output: "Remaining health: 56%"
```
6.1 Project setup

6.1.1 Project organization

Introduction

This tutorial is aimed to propose a simple workflow on how to organize projects. Since Godot allows the programmer to use the file-system as he or she pleases, figuring out a way to organize the projects when starting to use the engine can be a little challenging. Because of this, a simple workflow will be described, which can be used or not, but should work as a starting point.

Additionally, using version control can be challenging so this proposition will include that too.

Organization

Other game engines often work by having an asset database, where you can browse images, models, sounds, etc. Godot is more scene-based in nature so most of the time the assets are bundled inside the scenes or just exist as files but are referenced from scenes.

Importing & game folder

It is very often necessary to use asset importing in Godot. As the source assets for importing are also recognized as resources by the engine, this can become a problem if both are inside the project folder, because at the time of export the exporter will recognize them and export both.

To solve this, it is a good practice to have your game folder inside another folder (the actual project folder). This allows to have the game assets separated from the source assets, and also allows to use version control (such as svn or git) for both. Here is an example:

```
myproject/art/models/house.max
myproject/art/models/sometexture.png
myproject/sound/door_open.wav
myproject/sound/door_close.wav
myproject/translations/sheet.csv
```

Then also, the game itself is, in this case, inside a game/ folder:
Following this layout, many things can be done:

- The whole project is still inside a folder (myproject/).
- Exporting the project will not export the .wav and .png files which were imported.
- myproject/ can be put directly inside a VCS (like svn or git) for version control, both game and source assets are kept track of.
- If a team is working on the project, assets can be re-imported by other project members, because Godot keeps track of source assets using relative paths.

### Scene organization

Inside the game folder, a question that often arises is how to organize the scenes in the filesystem. Many developers try asset-type based organization and end up having a mess after a while, so the best answer is probably to organize them based on how the game works and not based on asset type. Here are some examples.

If you were organizing your project based on asset type, it would look like this:

```
game/engine.cfg
game/scenes/scene1.scn
game/scenes/scene2.scn
game/textures/texturea.png
game/textures/another.tex
game/sounds/sound1.smp
game/sounds/sound2.wav
game/music/music1.ogg
```

Which is generally a bad idea. When a project starts growing beyond a certain point, this becomes unmanageable. It's really difficult to tell what belongs to what.

It’s generally a better idea to use game-context based organization, something like this:

```
game/engine.cfg
game/scenes/house/house.scn
game/scenes/house/house.tex
game/scenes/valley/canyon.scn
game/scenes/valley/rock.scn
game/scenes/valley/rock.tex
game/scenes/common/tree.scn
game/scenes/common/tree.tex
game/player/player.scn
game/player/player.gd
game/npc/theking.scn
game/npc/theking.gd
game/gui/main_screen/main_screen.scn
```

This model or similar models allows projects to grow to really large sizes and still be completely manageable. Notice that everything is based on parts of the game that can be named or described, like the settings screen or the valley.
Since everything in Godot is done with scenes, and everything that can be named or described can be a scene, this workflow is very smooth and easygoing.

Cache files

Godot uses a hidden file called ".fscache" at the root of the project. On it, it caches project files and is used to quickly know when one is modified. Make sure to not commit this file to git or svn, as it contains local information and might confuse another editor instance in another computer.

6.2 Assets workflow

6.2.1 Import process

What is it for?

One of the most difficult things to get right when creating game engines is managing the import process. That means, getting the assets that artists make into the game, in a way that functions optimally.

Artists use certain tools and formats, and programmers would rather have their data in a different format. This is because artists put their focus on creating assets with the best quality possible, while programmers have to make sure they actually run at decent speed (or run at all), use a certain amount of memory, and don’t take ages loading from disk.

One would think that just writing a converter/importer would be enough, but this is not all there is to it. The same way programmers iterate several times over their code, artists keep making changes to their assets. This generates a bottleneck, because someone has to keep re-importing that artwork, right? And importing assets is often something that has to be agreed by both parties, as the programmer needs to decide how the artwork is imported and the artists needs to see how it looks.

The goal to establishing an import process is that both can agree on how the rules under which the assets are going to be imported the first time, and the system will apply those rules automatically each time the asset is re-imported.

Godot does not do the re-import process automatically, though. It gives the team the option to do it at any time (a red icon on the top right of the screen, allows the ability to do it at any desired time).

Does it always work?

The aim of the import system is that it works well enough for most common cases and projects. What is there has been tested and seems to cover most needs.

However, as mentioned before, this is one of the most difficult areas of writing a game engine. It may happen often (specially on large projects, ports, or projects with unusual requirement) that what is provided is not enough. It’s easy to say that the engine is open source and that the programmer should make their own if they don’t like what is there, but that would be making a huge disservice to the users and not the right attitude. Because of that, we made sure to provide as many tools and helpers as possible to support a custom import process, for example:

- Access to the internals of almost all data structures is provided to the scripting and C++ API, as well as saving and loading in all supported file formats.
- Some importers (like the 3D asset importer) support scripts to modify the data being imported.
- Support for creating custom import plugins is also provided, even for replacing the existing ones.
- If all else fails, Godot supports adding custom resource loaders, to load data in alternative formats, without intermediate conversion.
Both the import system and the custom tools provided will improve over time as more use cases are revealed to us.

**Importing assets**

**Source asset location**

To begin, it is a good idea to define where the original assets created by the artists (before they are imported) will be located. Normally, Godot does not mind much about the location, but if the project has several developers, it is a good idea to understand the simple rule for it to work for everyone.

First of all, it would be really good for this location to **not** be inside the project path (where engine.cfg is located, or any sub-folder). Godot expects regular resources in there, and may consider many of the files used as source art as regular resources. This would lead to it bundling all of them when the project is exported, something which is undesired.

Now that it is clear that this location must be outside the project folder, the rule that Godot uses to reference external assets can be explained. When an asset is imported, the engine stores a relative path from the project path to the asset (In windows, this works as long as they are on the same drive, otherwise an absolute path is stored). This ensures that the same asset can be re-imported in another computer.

The usual approach to this, when using a VCS such as Git, Mercurial or Subversion, is to create the project in a subfolder, so both the game’s project files and the source assets can be committed to a same repository. For example, the repository layout can look like this:

```plaintext
source_assets/sfx/explosion.wav
source_assets/sfx/crash.wav
source_assets/fonts/myfont.ttf
source_assets/translation/strings.csv
source_assets/art/niceart.psd
game/engine.cfg
```

In the above example, artists, musician, translators, etc. can work in the `source_assets/` folder, then import the assets to the `game/` folder. When the repository is updated, anyone can re-import the assets if they changed.

**Import dialogs**

Godot provides for importing several types of assets, all of them can be accessed from the import dialog:
Each of the dialogs shares a similar function, a source file (or several of them) must be provided, as well as a target destination inside the project folders. Once imported, Godot saves this information as metadata in the imported asset itself.
More information about each specific type of asset can be found in specific sections, such as Importing Textures.

**Tracking changes and re-importing**

Godot tracks changes in the source assets constantly. If at least one asset has been found to be modified (md5 is different than when it was imported), a small red indicator will appear in the top right corner of the screen.

From that moment onward, the user can choose to re-import at any given time by clicking on the red-icon. When this action is done, a dialog will pop-up showing which resources can be re-imported (all selected by default).
Accepting that dialog will immediately re-import the resources and will update any of them currently in use in the editor (like a texture, model or audio file).

**Manually re-importing**

The re-import process is automatic, but it may be desired at some point to change the settings of an already imported file, so it can be re-imported differently. For this, the Import Settings window is provided.
This screen allows the user to re-open the corresponding import-window to re-import that asset again, with the ability to change any of the settings.
### 6.2.2 Managing image files

If you have read the previous tutorials on Resources and File system, at this point you know that regular image files (.png, .jpg, etc.) are treated as regular resources in Godot.

Unlike texture resources (.tex files), image files contain no extra information on tiling (texture repeat), mipmaps or filtering. Editing this information and saving the texture back will not have any effect, since such formats can’t contain that information.

#### Image loader

Loading of images is done by the image loader. The behavior of the loader for all image files can be changed in the Project Settings dialog (Scene -> Project Settings). There is a section with values that are used for all image resources:

![Project Settings (engine.cfg) dialog](image_loader_options.png)

#### Image loader options

**Filter**

Filter is used when the image is stretched more than its original size, so a texel in the image is bigger than a pixel on the screen. Turning off the filter produces a retro-like look:
Repeat

Repeat is mainly used for 3D textures, so it’s off by default (textures are imported with the scenes and usually are not in the project as image files). When using UV coordinates (something not as common in 2D), and the UV value goes beyond the 0,0,1,1 rect, the texture repeats instead of clamping to the edge.

Mipmaps

When the mipmaps option is enabled, Godot will generate mipmaps. Mipmaps are versions of the image shrunk by half in both axes, recursively, until the image is 1 pixel of size. When the 3D hardware needs to shrink the image, it finds the largest mipmap it can scale from, and scales from there. This improves performance and image quality.

Alpha blending

The blending equation used by applications like Photoshop is too complex for real-time. There are better approximations such as pre-multiplied alpha, but they impose more stress in the asset pipeline. In the end, we are left with
textures that have artifacts in the edges, because apps such as Photoshop store white pixels in completely transparent areas. Such white pixels end up showing thanks to the texture filter (when active).

Godot has an option to fix the edges of the image (by painting invisible pixels the same color as the visible neighbours):

To do this, open the image from the resources tab, or edit it from the property editor from another node or resource, then go to the object options and select “Fix Alpha Edges”, then save it.
Since fixing this in so many images can be a little annoying, both Texture Import and Image Export can also perform this operation.

Texture import

Sometimes, it might be desirable to change the above settings per image. Unfortunately, the image loader settings are global. Texture flags also can’t be saved in a regular .png or .jpg file.

For such cases, the image can be imported as a texture (.tex), where the individual flags can be changed. Godot also
keeps track of the original file and will re-import if it changes.

Importing also allows conversion to other formats (WebP, or RAM compression) which might be of use in some cases. More information on the Importing textures page.

Image export

It is also possible to convert images to other formats (WebP or RAM compression) on export, as well as instructing the exporter to create an Atlas for a set of images. It is also possible to ask the exporter to scale all images (or selected groups).

More information on the Exporting images page.

Fixing PNGs iCCP chunk

With the upgrade of libpng to 1.6.23, libpng became more strict in terms of enforcing iCC profile correctness. This means that it now warns when it comes across an image with a non-conforming iCC chunk.

WARNING: _png_warn_function: iCCP: known incorrect sRGB profile

This can be fixed by either using a tool that exports PNGs with the correct iCC profile (in some tools this profile can even be manually changed on export) or using a tool that removes/fixes the iCC chunks.

Linux/Mac

Using ImageMagicks convert or mogrify fixes these warnings. To fix all PNGs in a project folder do:

```
$ find . -type f -name "*.png" -exec convert {} {} 
```

pngcheck is also useful in locating the non-conforming images:

```
find . -type f -name "*.png" -exec pngcheck {} 
```

Windows

Using optiPNG <http://optipng.sourceforge.net/> fixes these warnings on Windows. To fix a PNG inplace do:

```
optipng -clobber -strip all file.png
```

6.2.3 Importing textures

Do NOT import them in most cases

In most cases you don’t want images imported when dealing with 2D and GUI. Just copy them to the filesystem. Read the tutorial on Managing image files before continuing! For 3D, textures are always imported by the 3D scene importer, so importing those is only useful when importing a texture used for 3D that doesn’t come with the 3D scene (for example, in a shader). The flags and options are the same as here, so reading the rest of the document might help too.
OK, you might want to import them

So, if you have read the previous tutorial on the texture exporter, the texture importer gives you more fine-grained control on how textures are imported. If you want to change flags such as repeat, filter, mipmaps, fix edges, etc., importing them is the best way to accomplish this (since you can’t save such flags in a standard image file).

Lack of MipMaps

Images in 3D hardware are scaled with a (bi)linear filter, but this method has limitations. When images are shrunk too much, two problems arise:

- **Aliasing**: Pixels are skipped too much, and the image shows discontinuities. This decreases quality.
- **Cache Misses**: Pixels being read are too far apart, so texture cache reads a lot more data than it should. This decreases performance.

To solve this, mipmaps are created. Mipmaps are versions of the image shrunk by half in both axis, recursively, until the image is 1 pixel of size. When the 3D hardware needs to shrink the image, it finds the largest mipmap it can scale from, and scales from there. This improves performance and image quality.

Godot automatically creates mipmaps upon load for standard image files. This process is time consuming (although not much) and makes load times a little worse. Pre-importing the textures allows the automatic generation of mipmaps.

Unwanted MipMaps

Remember the previous point about mipmaps? Yes, they are cool, but mobile GPUs only support them if the textures are in power of 2 dimensions (i.e. 256x256 or 512x128). In these platforms, Godot will stretch and enlarge the texture to the closest power of 2 size and then generate the mipmaps. This process takes more of a performance hit and it might degrade the quality a little more.
Because of this, there are some scenarios when it may be desirable to not use them, and just use a linear filter. One of them is when working with graphical user interfaces (GUIs). Usually they are made of large images and don’t stretch much. Even if the screen resolution is in a larger or smaller value than original art, the amount of stretch is not as much and the art can retain the quality. Pre-importing the textures also allows the disabling of mipmap generation.

**Blending artifacts**

The blending equation used by applications like Photoshop is too complex for realtime. There are better approximations such as pre-multiplied alpha, but they impose more stress in the asset pipeline. In the end, we are left with textures that have artifacts in the edges, because apps such as Photoshop store white pixels in completely transparent areas. Such white pixels end up showing thanks to the texture filter.

Godot has an option to fix the edges of the image (by painting invisible pixels the same color as the visible neighbours):

![Photoshop vs Fixed in Godot](image)

However, this must be done every time the image changes. Pre-Importing the textures makes sure that every time the original file changes, this artifact is fixed upon automatic re-import.

**Texture flags**

Textures have flags. The user can choose for them to repeat or clamp to edges (when UVs exceed the 0,0,1,1 boundary). The magnifying filter can also be turned off (for a Minecraft-like effect). Such values can not be edited in standard file formats (png, jpg, etc.), but can be edited and saved in Godot .tex files. Then again, the user may not want to change the values every time the texture changes. Pre-Importing the textures also takes care of that.

**Texture compression**

Aside from the typical texture compression, which saves space on disk (.png, .jpg, etc.), there are also texture compression formats that save space in memory (more specifically video memory. This allows to have much better looking textures in games without running out of memory, and decrease memory bandwidth when reading them so they are a big plus.

There are several video texture compression formats, none of which are standard. Apple uses PVRTC. PC GPUs, consoles and nVidia Android devices use S3TC (BC), other chipsets use other formats. OpenGL ES 3.0 standardized on ETC format, but we are still a few years away from that working everywhere.

Still, when using this option, Godot converts and compresses to the relevant format depending on the target platform (as long as the user pre-imported the texture and specified video ram compression!).

This kind of compression is often not desirable for many types of 2D games and UIs because it is lossy, creating visual artifacts. This is especially noticeable on games that use the trendy vectory social game artwork. However, the fact that it saves space and improves performance may make up for it.

The 3D scene importer always imports textures with this option turned on.
Atlases

Remember how mobile GPUs have this limitation of textures having to be in power of 2 sizes to be able to generate mipmap maps for optimum stretching? What if we have a lot of images in different random sizes? All will have to be scaled and mipmap mapped when loaded (using more CPU and memory) or when imported (taking more storage space). This is probably still OK, but there is a tool that can help improve this situation.

Atlases are big textures that fit a lot of small textures inside efficiently. Godot supports creating atlases in the importer, and the imported files are just small resources that reference a region of the bigger texture.

Atlases can be a nice solution to save some space on GUI or 2D artwork by packing everything together. The current importer is not as useful for 3D though (3D Atlases are created differently, and not all 3D models can use them).

As a small plus, atlases can decrease the amount of “state changes” when drawing. If a lot of objects that are drawn using several different textures are converted to an atlas, then the texture rebinds per object will go from dozens or hundreds to one. This will give the performance a small boost.

Artists use PSD

Still wondering whether to use the texture importer or not? Remember that in the end, artists will often use Photoshop anyway, so it may be wiser to just let the import subsystem to take care of importing and converting the PSD files instead of asking the artist to save a png and copy it to the project every time.

Texture importer

Finally! It’s time to take a look at the texture importer. There are 3 options in the import menu. They are pretty much (almost) the same dialog with a different set of defaults.

When selected, the texture import dialog will appear. This is the default one for 2D textures:
Each import option has a function, explained as follows:

**Source texture(s)**

One or more source images can be selected from the same folder (this importer can do batch-conversion). This can be from inside or outside the project.

**Target path**

A destination folder must be provided. It must be inside the project, as textures will be converted and saved to it. Extensions will be changed to .tex (Godot resource file for textures), but names will be kept.

**Texture format**

This combo allows to change the texture format (compression in this case):

- Uncompressed
- Compress Lossless (PNG)
- Compress Lossy (WebP)
- Compress (VRAM)

Each of the four options described in this table together with their advantages and disadvantages (่อน = Best, ≈Worst):
<table>
<thead>
<tr>
<th>Description</th>
<th>Uncompressed</th>
<th>Compress Lossless (PNG)</th>
<th>Compress Lossy (WebP)</th>
<th>Compress VRAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size on Disk</td>
<td>Large</td>
<td>Small</td>
<td>Very Small</td>
<td>Small</td>
</tr>
<tr>
<td>Memory Usage</td>
<td>Large</td>
<td>Large</td>
<td>Large</td>
<td>Small</td>
</tr>
<tr>
<td>Performance</td>
<td>Normal</td>
<td>Normal</td>
<td>Normal</td>
<td>Fast</td>
</tr>
<tr>
<td>Quality Loss</td>
<td>None</td>
<td>None</td>
<td>Slight</td>
<td>Moderate</td>
</tr>
<tr>
<td>Load Time</td>
<td>Normal</td>
<td>Slow</td>
<td>Slow</td>
<td>Fast</td>
</tr>
</tbody>
</table>

### Texture options

Provided are a small amount of options for fine grained import control:

- **Streaming Format** - This does nothing as of yet, but a texture format for streaming different mipmap levels is planned. Big engines have support for this.

- **Fix Border Alpha** - This will fix texture borders to avoid the white auras created by white invisible pixels (see the rant above).

- **Alpha Bit Hint** - Godot auto-detects if the texture needs alpha bit support for transparency (instead of full range), which is useful for compressed formats such as BC. This forces alpha to be 0 or 1.

- **Compress Extra** - Some VRAM compressions have alternate formats that compress more at the expense of quality (PVRTC2 for example). If this is ticked, texture will be smaller but look worse.

- **No MipMaps** - Force imported texture to NOT use mipmaps. This may be desirable in some cases for 2D (as explained in the rant above), though it’s NEVER desirable for 3D.

- **Repeat** - Texture will repeat when UV coordinates go beyond 1 and below 0. This is often desirable in 3D, but may generate artifacts in 2D.

- **Filter** - Enables linear filtering when a texture texel is larger than a screen pixel. This is usually turned on, unless it’s required for artistic purposes (Minecraft look, for example).

### 6.2.4 Importing audio samples

#### Why importing?

Importing Audio Samples into the game engine is a process that should be easier than it really is. Most readers are probably thinking “Why not just copy the wav files to a folder inside the project and be over with it?”

It’s not usually that simple. Most game engines use uncompressed audio (in memory, at least) for sound effects. The reason for this is because it’s really cheap to play back and resample. Compressed streamed audio (such as ogg files) takes a large amount of processor to decode so no more than one or two are streamed simultaneously. However, with sound effects, one expects a dozen of them to be playing at the same time in several situations.

Because of this, sound effects are loaded uncompressed into memory, and here is where the problems begin.
As is usual with graphics, the situation where programmers don’t really know about audio and audio engineers don’t know about programming is also common in the industry. This leads to a scenario where a project ends up wasting resources unnecessarily.

To be more precise, SFX artists tend to work with audio formats that give them a lot of room for tweaking the audio with a low noise floor and minimum aliasing, such as 96kHz, 24 bits. In many cases, they work in stereo too. Added to that, many times they add effects with an infinite or really long fadeout, such as reverb, which leads to apparent trailing silences. Finally, many DAWs also add silence at the beginning when normalizing to wav.

These often result in extremely large files to integration into a game engine with sound effects taking dozens of megabytes.

**How much does quality matter?**

First of all, it is important to know that Godot has an internal reverb generator. Sound effects can go to four different setups (small, medium and large room, as well as hall), with different send amounts. This saves SFX artists the need to add reverb to the sound effects, reducing their size greatly and ensuring correct trimming. Say no to SFX with baked reverb!
Another common problem is that, while it's useful for working inside a DAW, high bit depths (24 bits) and high sampling rate (96kHz) are completely unnecessary for use in a game, as there is no audible difference. If positional sound is going to be used (for 2D and 3D), the panning and stereo reverb will be provided by the engine, so there is little need for stereo sound. How does this affect the resource usage? Look at the following comparison:

<table>
<thead>
<tr>
<th>Format</th>
<th>1 Second of Audio</th>
<th>Frame Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>24 bits, 96 kHz, Stereo</td>
<td>576kb</td>
<td>12</td>
</tr>
<tr>
<td>16 bits, 44 kHz, Mono</td>
<td>88kb</td>
<td>2</td>
</tr>
<tr>
<td>16 bits, IMA-ADPCM</td>
<td>22kb</td>
<td>1/2</td>
</tr>
</tbody>
</table>

As seen, for being no audible difference, the 16 bits, 44kHz, mono conversion takes 6 times less memory than the 24 bits, 96kHz, Stereo version. The IMA-ADPCM version (using computationally-light audio compression) takes 24 times less memory than what was exported from the DAW.
Trimming

One last issue that happens often is that the waveform files received have silences at the beginning and at the end. These are inserted by DAWs when saving to a waveform, increase their size unnecessarily and add latency to the moment they are played back. Trimming them solves this, but it takes effort for the SFX artist, as they have to do it in a separate application. In the worst case, they may not even know the silences are being added.

Importing audio samples

Godot has a simple screen for importing audio samples to the engine. SFX artists only have to save the wav files to a folder outside the project, and the import dialog will fix the files for inclusion, as well as doing it automatically every time they are modified and re-imported.
In this screen, the quality of the audio can be limited to what is needed, and trimming is done automatically. In addition, several samples can be loaded and batch-converted, just as textures can.

**Looping**

Godot supports looping in the samples (Tools such as Sound Forge or Audition can add loop points to wav files). This is useful for sound effects such as engines, machine guns, etc. Ping-pong looping is also supported.

As an alternative, the import screen has a “loop” option that enables looping for the entire sample when importing.

### 6.2.5 Importing fonts

**What is a font?**

Fonts in modern operating systems are created as scalable vector graphics. They are stored as a collection of curves (usually one for each character), which are independent of the screen resolution, and stored in standardized file formats,
such as TTF (TrueType) or OTF (OpenType).

Rendering such fonts to bitmaps is a complex process, which employs different methods to convert curves to pixels depending on context and target size. Due to this, this rendering process must be done by using the CPU. Game engines use the GPU to render, and 3D APIs don’t really support the means to do this efficiently, so fonts have to be converted to a format that is friendly to the GPU when imported to a project.

Converting fonts

This conversion process consists of rendering a vector font to a given point size and storing all the resulting characters in a bitmap texture. The bitmap texture is then used by the GPU to draw a small quad for each character and form readable strings.

The drawback of this process is that fonts must be pre-imported in the specific sizes that they will use in the project. However, given that that bitmap fonts compress really well, this is not as bad as it sounds.

Importing a font

Fonts are imported via the Font import dialog. The dialog will ask for a font, a size, some options and a target resource file to save.
The dialog is fully dynamic, which means that any change will be reflected in the font preview window. The user can tweak almost every parameter and get instant feedback on how the font will look.

Since the resulting font is a bitmap, a few more options were added to make the imported font look even nicer. These options were added to please graphic designers, who love putting gradients, outlines and shadows in fonts, as well as changing all the inter-spaces available :). These options will be explained in the next section.

**Extra spacing**

It is possible to add more space for:

- **Characters**, the space between them can be varied.
- **“space” character**, so the distance between words is bigger.
- **Top and Bottom margins**, this changes the spacing between lines as well as the space between the top and bottom lines and the borders.
Shadows & outline

Fonts can have a shadow. For this, the font is drawn again, below the original, in a different color, and then blurred with a Gaussian kernel of different sizes. The resulting shadow can be adjusted with an exponential function to make it softer or more like an outline. A second shadow is also provided to create some added effects, like a bump or outline+shadow.

Gradients

Gradients are also another of the visual effects that graphic designers often use. To show how much we love them, we added those too. Gradients can be provided as a simple curve between two colors, or a special png file with a hand drawn gradient.

Internationalization

Colors, shadows and gradients are beautiful, but it’s time we get to serious business. Developing games for Asian markets is a common practice in today’s globalized world and app stores.

Here’s when things get tricky with using bitmap fonts. Asian alphabets (Chinese, Japanese and Korean) contain dozens of thousands of characters. Generating bitmap fonts with every single of them is pretty expensive, as the resulting textures are huge. If the font size is small enough, it can be done without much trouble, but when the fonts become bigger, we run out of video ram pretty quickly!

To solve this, Godot allows the user to specify a text file (in UTF-8 format) where it expects to find all the characters that will be used in the project. This seems difficult to provide at first, and more to keep up to date, but it becomes rather easy when one realizes that the .csv with the translations can be used as such a source file (see the Importing translations section). As Godot re-imports assets when their dependencies change, both the translation and font files will be updated and re-imported automatically if the translation csv changes.

Another cool trick for using a text file as limit of which characters can be imported is when using really large fonts. For example, the user might want to use a super large font, but only to show numbers. For this, he or she writes a numbers.txt file that contains “1234567890”, and Godot will only limit itself to import data, thus saving a lot of video memory.
6.2.6 Importing translations

Games and internationalization

The world is full of different markets and cultures and, to maximize profits™, nowadays games are released in several languages. To solve this, internationalized text must be supported in any modern game engine.

In regular desktop or mobile applications, internationalized text is usually located in resource files (or .po files for GNU stuff). Games, however, can use several orders of magnitude more text than applications, so they must support efficient methods for dealing with loads of multilingual text.

There are two approaches to generate multilingual language games and applications. Both are based on a key:value system. The first is to use one of the languages as the key (usually English), the second is to use a specific identifier. The first approach is probably easier for development if a game is released first in English, later in other languages, but a complete nightmare if working with many languages at the same time.

In general, games use the second approach and a unique ID is used for each string. This allows you to revise the text while it is being translated to other languages. The unique ID can be a number, a string, or a string with a number (it’s just a unique string anyway).

Translators also usually prefer to work with spreadsheets.

Translation format

To complete the picture and allow efficient support for translations, Godot has a special importer that can read CSV files. All spreadsheet editors (be it Libreoffice, Microsoft Office, Google Docs, etc.) can export to this format, so the only requirement is that the files have a special arrangement. The CSV files must be saved in UTF-8 encoding and be formatted as follows:

```
<table>
<thead>
<tr>
<th>&lt;lang1&gt;</th>
<th>&lt;lang2&gt;</th>
<th>&lt;langN&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>KEY1</td>
<td>string</td>
<td>string</td>
</tr>
<tr>
<td>KEY2</td>
<td>string</td>
<td>string</td>
</tr>
<tr>
<td>KEYN</td>
<td>string</td>
<td>string</td>
</tr>
</tbody>
</table>
```

The “lang” tags must represent a language, which must be one of the valid locales supported by the engine. The “KEY” tags must be unique and represent a string universally (they are usually in uppercase, to differentiate from other strings). Here’s an example:

```
id | en     | es     | ja     |
---|--------|--------|--------|
GREET | Hello, friend! | Hola, Amigo! |
ASK   | How are you?   | Cómo estás?  |
BYE   | Good Bye       | Adiós       |
```

Import dialog

The import dialog takes a CSV file in the previously described format and generates several compressed translation resource files inside the project.

Selecting a CSV file autodetects the languages from the first row and determines which column represents which language. It is possible to change this manually, by selecting the language for each column.
The import dialog also can add the translation to the list of translations to load when the game runs, specified in engine.cfg (or the project properties). Godot allows loading and removing translations at runtime as well.

6.2.7 Exporting images

It is often desirable to perform an operation on all or a group of images upon export. Godot provides some tools for this. Examples of such operations are:

- Converting all images from a lossless format to a lossy one (ie: png -> WebP) for greater compression.
- Shrinking all images to half the size, to create a low resolution build for smaller screens.
- Create an atlas for a group of images and crop them, for higher performance and less memory usage.

Image export options

In the “Project Export Settings” dialog, go to the Images tab:
In this dialog the image extensions for conversion can be selected, and operations can be performed that apply to all images (except those in groups, see the next section for those):

- **Convert Image Format**: Probably the most useful operation is to convert to Lossy (WebP) to save disk space. For lossy, a Quality bar can set the quality/vs size ratio.

- **Shrink**: This allows to shrink all images by a given amount. It’s useful to export a game to half or less resolution for special devices.

- **Compress Formats**: Allows to select which image extensions to convert.

On export, Godot will perform the desired operation. The first export might be really slow, but subsequent exports will be fast, as the converted images will be cached.

### Image group export options

This section is similar to the previous one, except it can operate on a selected group of images. When a image is in a group, the settings from the global export options are overridden by the ones from the group. An image can only be in one group at a time. So if the image is in a group other than the current one being edited, it will not be selectable.
Atlas

Grouping images allows a texture atlas to be created. When this mode is active, a button to preview the resulting atlas becomes available. Make sure that atlases don’t become too big, as some hardware will not support textures bigger than 2048x2048 pixels. If this happens, just create another atlas.

The atlas can be useful to speed up drawing of some scenes, as state changes are minimized when drawing from it (though unlike other engines, Godot is designed so state changes do not affect it as much). Textures added to an atlas get cropped (empty spaces around the image are removed), so this is another reason to use them (save space). If unsure, though, just leave that option disabled.

6.3 Export

6.3.1 Exporting projects

Why exporting?

Originally, Godot did not have any means to export projects. The developers would compile the proper binaries and build the packages for each platform manually.
When more developers (and even non-programmers) started using it, and when our company started taking more projects at the same time, it became evident that this was a bottleneck.

**On PC**

Distributing a game project on PC with Godot is rather easy. Just drop the godot.exe (or godot) binary together in the same place as the engine.cfg file, zip it and you are done. This can be taken advantage of to make custom installers.

It sounds simple, but there are probably a few reasons why the developer may not want to do this. The first one is that it may not be desirable to distribute loads of files. Some developers may not like curious users peeking at how the game was made, others may just find it inelegant, etc.

Another reason is that, for distribution, the developer might prefer a specially compiled binary, which is smaller in size, more optimized and does not include tools inside (like the editor, debugger, etc.).

Finally, Godot has a simple but efficient system for creating DLCs as extra package files.

**On mobile**

The same scenario in mobile is a little worse. To distribute a project in those devices, a binary for each of those platforms is built, then added to a native project together with the game data.

This can be troublesome because it means that the developer must be familiarized with the SDK of each platform before even being able to export. While learning each SDK is always encouraged, it can be frustrating to be forced to do it at an undesired time.

There is also another problem with this approach. Different devices prefer some data in different formats to run. The main example of this is texture compression. All PC hardware uses S3TC (BC) compression and that has been standardized for more than a decade, but mobile devices use different formats for texture compression, such as PVRCT (iOS) or ETC (Android).

**Export dialog**

After many attempts at different export workflows, the current one has proven to work the best. At the time of this writing, not all platforms are supported yet, but the supported platforms continue to grow.

To open the export dialog, just click the “Export” button:

![Scene Import Export](image)

The dialog will open, showing all the supported export platforms:
The default options are often enough to export, so tweaking them is not necessary, but provide extra control. However, many platforms require additional tools (SDKs) to be installed to be able to export. Additionally, Godot needs export templates installed to create packages. The export dialog will complain when something is missing and will not allow the user to export for that platform until they resolve it:

![Export dialog](image)

At that time, the user is expected to come back to the documentation and follow instructions on how to properly set up that platform.

**Export templates**

Apart from setting up the platform, the export templates must be installed to be able to export projects. They can be obtained as a .tpz (a renamed .zip) file from the download page of the website.

Once downloaded, they can be installed using the “Install Export Templates” option under the “Settings” menu in the top right corner of the editor:
Export mode

When exporting, Godot makes a list of all the files to export and then creates the package. There are 3 different modes for exporting:

- Export every single file in the project
- Export only resources (+custom filter), this is default.
- Export only selected resources (+custom filter)

- **Export every single file** - This mode exports every single file in the project. This is good to test if something is being forgotten, but developers often have a lot of unrelated stuff around in the dev directory, which makes it a bad idea.

- **Export only resources** - Only resources are exported. For most projects, this is enough. However many
developers like to use custom datafiles in their games. To compensate for this, filters can be added for extra extensions (like,.txt,.csv, etc.).

- **Export only selected resources** - Only select resources from a list are exported. This is probably overkill for most projects, but in some cases it is justified (usually huge projects). This mode offers total control of what is exported. Individual resources can be selected and dependency detection is performed to ensure that everything needed is added. As a plus, this mode allows to “Bundle” scenes and dependencies into a single file, which is really useful for games distributed on optical media.

### 6.3.2 Exporting for PC

The simplest way to distribute a game for PC is to copy the executables (godot.exe on windows, godot on the rest), zip the folder and send it to someone else. However, this is often not desired.

Godot offers a more elegant approach for PC distribution when using the export system. When exporting for PC (Linux, Windows, Mac), the exporter takes all the project files and creates a “data.pck” file. This file is bundled with a specially optimized binary that is smaller, faster and lacks tools and debugger.

Optionally, the files can be bundled inside the executable, though this does not always works properly.

### 6.3.3 Exporting for iOS

Exporting for iOS is done manually at the moment. These are the steps to load your game in an XCode project, where you can deploy to a device, publish, etc.
Requirements

- Download XCode for iOS
- Download the export templates: https://godotengine.org/download
- Since there is no automatic deployer yet, unzip export_templates.tpz manually and extract GodotiOSXCode.zip from it.

The zip contains an XCode project, godot_ios.xcodeproj, an empty data.pck file and the engine executable. Open the project, and modify the game name, icon, organization, provisioning signing certificate identities (??), etc.

Add your project data

Using the Godot editor, Exporting for PC, to obtain the data.pck file. Replace the empty data.pck in the XCode project with the new one, and run/archive.

If you want to test your scenes on the iOS device as you edit them, you can add your game directory to the project (instead of data.pck), and add a property “godot_path” to Info.plist, with the name of your directory as its value.

Alternatively you can add all the files from your game directly, with “engine.cfg” at the root.

Loading files from a host

Sometimes your game becomes too big and deploying to the device takes too long every time you run. In that case you can deploy only the engine executable, and serve the game files from your computer.

Setting up the file host

On your PC, open the editor, and click the right-most icon on the top-center group of icons, and select “Enable File Server”. The icon turns red. Your PC will open a port and accept connections to serve files from your project’s directory (so enable your local firewall accordingly).
Setting up the game

On XCode, click on your app name (top left, next to the “Stop” button), and select “Edit Scheme”. Go to the “Arguments” tab, and add 2 arguments, “-rfs” and the IP of your PC.

When you run, your device will connect to the host and open the files remotely. Note that the directory with the game data (“platformer”) is no longer added to the project, only the engine executable.

Services for iOS

Special iOS services can be used in Godot. Check out the Services for iOS page.

6.3.4 Exporting for Universal Windows Platform

There’s no extra requirement to export an .appx package that can be installed as a Windows App or submitted to the Windows Store. Exporting packages also works from any platform, not only on Windows.
However, if you want to install and run the app, you need to sign it with a trusted signature. Currently, Godot supports no signing of packages and you need to use externals to tools to do so.

Also, make sure the Publisher name you set when export the package matches the name on the certificate.

**Limitations on Xbox One**

As described in UWP documentation:

- available RAM is 1GB (after exceeding it, application will encounter memory allocation failures and will crash)
- share of 2-4 CPU cores
- share of 45% of GPU power

**Creating a signing certificate**

This requires the tools `MakeCert.exe` and `Pvk2Pfx.exe` which comes with the Windows SDK. If you use Visual Studio, open one of its Developer Prompts since they come with those tools available and in the path.

You can get more detailed instructions from Microsoft documentation.

First, run `MakeCert` to create a private key:

```
MakeCert /n publisherName /r /h 0 /eku "1.3.6.1.5.5.7.3.3,1.3.6.1.4.1.311.10.3.13" /e expirationDate /sv MyKey.pvk MyKey.cer
```

Where `publisherName` matches the Publisher Name of your package and `expirationDate` is in the `mm/dd/yyyy` format.

Next, create a Personal Information Exchange (.pfx) file using `Pvk2Pfx.exe`:

```
Pvk2Pfx /pvk MyKey.pvk /pi pvkPassword /spc MyKey.cer /pfx MyKey.pfx [/po pfxPassword]
```

If you don’t specify a password with `/po` argument, the PFX will have the same password as the private key.

You also need to trust this certificate to be able to actually install the apps. Open the Command Prompt as Administrator and run the following command:

```
Certutil -addStore TrustedPeople MyKey.cer
```

**Signing the package**

Using the `SignTool.exe` this requires a single command:

```
SignTool sign /fd SHA256 /a /f MyKey.pfx /p pfxPassword package.appx
```

**Installing the package**

After Windows 10 Anniversary Update you can install packages by just double clicking the `.appx` file from the Windows Explorer.

It’s also possible to install using the `Add-AppxPackage` PowerShell cmdlet.

Note that if you don’t update the version number, you’ll have to uninstall the previous installed package before reinstalling it.
6.3.5 Exporting for Android

Exporting for Android has fewer requirements than compiling Godot for it. The following steps detail what is needed to setup the SDK and the engine.

Download the Android SDK

Download and install the Android SDK from http://developer.android.com/sdk/index.html

Install OpenJDK or Oracle JDK

Download and install OpenJDK or Oracle JDK. Version 6 and 8 are known to work, some users have reported issues with the jarsigner (used to sign the APKs) in JDK 7.

Create a debug.keystore

Android needs a debug keystore file to install to devices and distribute non-release APKs. If you have used the SDK before and have built projects, ant or eclipse probably generated one for you (In Linux and OSX, you can find it in the ~/.android folder).

If you can’t find it or need to generate one, the keytool command from the JDK can be used for this purpose:

```
```

Make sure you have adb

Android Debug Bridge (adb) is the command line tool used to communicate with Android devices. It’s installed with the SDK, but you may need to install one (any) of the Android API levels for it to be installed in the SDK directory.

Setting it up in Godot

Enter the Editor Settings screen. This screen contains the editor settings for the user account in the computer (It’s independent from the project).

Scroll down to the section where the Android settings are located:
In that screen, the path to 3 files needs to be set:

- The *adb* executable (adb.exe on Windows)
- The *jarsigner* executable (from JDK 6 or 8)
- The debug *keystore*

Once that is configured, everything is ready to export to Android!

### 6.3.6 Exporting for the Web

Exporting for the web generates several files to be served from a web server, including a default HTML page for presentation. A custom HTML file can be used, see *Compiling for the Web*.

The default HTML file is designed to fit the game perfectly without cutting off parts of the canvas when the browser window is scaled to the game’s dimensions. This way it can be inserted into an `<iframe>` with the game’s size, as is common on most web game hosting sites.

Web export uses *asm.js*, a highly optimizable subset of JavaScript. A 64-bit browser is required to run games in asm.js format. Most notably, this is a problem with Firefox, which on Windows is shipped as a 32-bit application by default.

### Serving the files

The default `.html` file can be used as `DirectoryIndex` and can be renamed to e.g. `index.html` at any time, its name is never depended on. It can also be inserted into another HTML file as an `<iframe>` element. Users must allow *third-party* cookies when playing a game presented in an iframe.
The other exported files are served as they are next to the .html file, names unchanged.
The .mem and .pck files are binary, usually delivered with the MIME-type application/octet-stream.
Delivering the files with gzip compression is recommended especially for the .pck, .asm.js and .mem files, which are usually large in size.

Export options

Turning on Debugging Enabled when exporting will, in addition to enabling various debug features of the engine, display a debug output below the canvas, displaying JavaScript and engine errors. If controls are enabled as well, display of this output can be toggled. You can also use the browser-integrated developer console, usually opened with the F12 key, which often shows more information, including WebGL errors.

Memory Size is fixed and must thus be set during export. Try using no more than necessary to strain users’ browsers as little as possible.

Enable Run will add a button between the Stop scene and Play edited Scene buttons in the editor to quickly open the game in the default browser for testing.

The remaining options customize the generated HTML page:

Title is the content of the <title> element of the page, usually used by browsers as the tab and window name. The title set here is only displayed until the game is started, afterwards the title is set to the application name set in the project settings.

Head Include and Style Include are appended into the <head> and CSS <style> elements respectively. This allows, for example, linking web fonts for use in the page.

Font Family is the CSS font-family used on the page, without terminating semicolon.

Controls Enabled toggles display of controls, offering e.g. a toggle for output display in debug mode and a fullscreen button. In the default page, the controls are displayed in the top-right corner on top of the canvas, which can get in the way in games that use the cursor.

Web export limitations

Exported files must not be reused

The exported files ending with .html and fs.js are adjusted on export specifically for that game’s version and the given export options. They must not be reused in further exports.

Some functions must be called from input callbacks

Browsers do not allow arbitrarily entering full screen at any time. The same goes for capturing the cursor. Instead, these actions have to occur as a response to a JavaScript input event. In Godot, this is most easily done by entering full screen from within an input callback such as _input or _unhandled_input.

Starting exported games from the local file system

Many browsers will not load exported projects when opened locally per file:// protocol. To get around this, use a local server.

Python offers an easy method for this, using python -m SimpleHTTPServer with Python 2 or python -m http.server with Python 3 will serve the current working directory on http://localhost:8000.
Locale lookup

Godot tries to detect the user’s locale using information provided by the browser, but this is rather unreliable. A better way is to use CGI to read the HTTP Accept-Language header. If you assign its value to the JavaScript property Module.locale after the Module objects is created, but before the engine starts, Godot will use that value to initialize the locale. In any case, users should always be offered the option to configure the locale manually.

6.3.7 One-click deploy

Sounds good, what is it?

This feature will pop up automatically once a platform is properly configured and a supported device is connected to the computer. Since things can go wrong at many levels (platform may not be configured correctly, SDK may incorrectly installed, device may be improperly configured, kitty ate the USB cable, etc.), it’s good to let the user know that it exists.

Some platforms (at the time of this writing, only Android and Blackberry 10) can detect when a USB device is connected to the computer, and offer the user to automatically export, install and run the project (in debug mode) on the device. This feature is called, in industry buzz-words, “One Click Deploy” (though, it’s technically two clicks...).

Steps for one-click deploy

1. Configure target platform.

2. Configure device (make sure it’s in developer mode, likes the computer, usb is recognized, usb cable is plugged, etc.).

3. Connect the device..

4. And voila!

Click once.. and deploy!
7.1 Making Plugins

*Important:* This tutorial applies only to the upcoming version 2.1.

7.1.1 About Plugins

A plugin is a great way to extend the editor with useful tools. It can be made entirely with GDScript and standard scenes, without even reloading the editor. Unlike modules, you don’t need to create C++ code nor recompile the engine. While this makes plugins not as powerful, there’s still a lot of things you can do with them. Note that a plugin is not different from any scene you already can make, except that it is made via script to add functionality.

This tutorial will guide you through the creation of two simple plugins so you can understand how they work and be able to develop your own. The first will be a custom node that you can add to any scene in the project and the other will be a custom dock added to the editor.

7.1.2 Creating a plugin

Before starting, create a new empty project wherever you want. This will serve as base to develop and test the plugins.

The first thing you need to do is to create a new plugin that the editor can understand as such. For that you need two files: `plugin.cfg` for the configuration and a custom GDScript with the functionality.

Plugins have a standard path like `addons/plugin_name` inside the project folder. So create the folder `my_custom_node` inside `addons`. So you’ll have a directory structure like this:
To make the plugin.cfg file, open your favorite text editor with a blank file. Godot is not able (yet) to open text files besides scripts, so this must be done in an external editor. Add the following structure to your plugin.cfg:

```
[plugin]
name="My Custom Node"
description="A custom node made to extend the Godot Engine."
author="Your Name Here"
version="1.0"
script="custom_node.gd"
```

This is a simple ini file with metadata about your plugin. You need to set up the name and description so users can understand what it does. Add your own name so you can be properly credited. A version number is useful so users can see if they have an outdated version (if you are unsure on how to come up with the version number, check SemVer). And finally a main script file to load when your plugin is active.

### The script file

Open the script editor (F3) and create a new GDScript file called `custom_node.gd` inside the `my_custom_node` folder. This script is special and it has two requirements: it must be a tool script and it has to inherit from `Editor-Plugin`.

It’s important to deal with initialization and clean-up of resources. So a good practice is to use the virtual function `_enter_tree()` to initialize your plugin and `_exit_tree()` to clean it up. You can delete the default GDScript template from your file and replace it with the following structure:

```
tool
extends EditorPlugin

func _enter_tree():
    # Initialization of the plugin goes here
    pass

func _exit_tree():
    # Clean-up of the plugin goes here
    pass
```

This is a good template to use when devising new plugins.
7.1.3 A custom node

Sometimes you want a certain behavior in many nodes. Maybe a custom scene or control that can be reused. Instancing is helpful in a lot of cases but sometimes it can be cumbersome, especially if you’re using it between many projects. A good solution to this is to make a plugin that adds a node with a custom behavior.

To create a new node type, you can avail of the function `add_custom_type()` from the `EditorPlugin` class. This function can add new types to the editor, be it nodes or resources. But before you can create the type you need a script that will act as the logic for the type. While such script doesn’t need to have the `tool` keyword, it is interesting to use it so the user can see it acting on the editor.

For this tutorial, we’ll create a simple button that prints a message when clicked. And for that we’ll need a simple script that extends from `Button`. It could also extend `BaseButton` if you prefer:

```gd
@extends Button

func _enter_tree():
    connect("pressed", self, "clicked")

func clicked():
    print("You clicked me!")
```

That’s it for our basic button. You can save this as `button.gd` inside the plugin folder. You’ll also need a 16x16 icon to show in the scene tree. If you don’t have one, you can grab the default one from the engine:

Now we need to add it as a custom type so it shows on the Create New Node dialog. For that, change the `custom_node.gd` script to the following:

```gd
@extends EditorPlugin

func _enter_tree():
    # Initialization of the plugin goes here
    # Add the new type with a name, a parent type, a script and an icon
    add_custom_type("MyButton", "Button", preload("button.gd"), preload("icon.png"))

func _exit_tree():
    # Clean-up of the plugin goes here
    # Always remember to remove it from the engine when deactivated
    remove_custom_type("MyButton")
```

With that done, the plugin should already be available in the plugin list at Project Settings. So activate it and try to add a new node to see the result:
When you add the node, you can see that it already have the script you created attached to it. Set a text to the button, save and run the scene. When you click the button, you can see a text in the console:

A custom dock

Maybe you need to extend the editor and add tools that are always available. An easy way to do it is to add a new dock with a plugin. Docks are just scenes based on control, so how to create them is not far beyond your knowledge.

The way to start this plugin is similar to the custom node. So create a new plugin.cfg file in the addons/my_custom_dock folder. And then with your favorite text editor add the following content to it:

```
[plugin]
name="My Custom Dock"
description="A custom dock made so I can learn how to make plugins."
author="Your Name Here"
version="1.0"
script="custom_dock.gd"
```

Then create the script custom_dock.gd in the same folder. Fill with the template we’ve seen before to get a good start.

Since we’re trying to add a new custom dock, we need to create the contents of such dock. This is nothing more than a standard Godot scene. So you can create a new scene in the editor and start creating it.

For an editor dock, it is mandatory that the root of the scene is a Control or one of its child classes. For this tutorial, you can make a single button. The name of the root node will also be the name that appears on the dock tab, so be sure to put a descriptive but short one. Don’t forget to add a text to your button.
Save this scene as `my_dock.tscn`.

Now you need to grab that scene you just created and add it as a dock in the editor. For this you can rely on the function `add_control_to_dock()` from the `EditorPlugin` class.

The code is very straightforward, you just need to select a dock position to add it and have a control to add (which is the scene you just created). It is also very important that you remember to remove the dock when the plugin is deactivated. The code can be like this:

```godot
tool
extends EditorPlugin

var dock # A class member to hold the dock during the plugin lifecycle

func _enter_tree():
    # Initialization of the plugin goes here
    # First load the dock scene and instance it:
    dock = preload("res://addons/my_custom_dock/my_dock.tscn").instance()

    # Add the loaded scene to the docks:
    add_control_to_dock( DOCK_SLOT_LEFT_UL, dock)
    # Note that LEFT_UL means the left of the editor, upper-left dock

func _exit_tree():
    # Clean-up of the plugin goes here
    # Remove the scene from the docks:
    remove_control_from_docks( dock ) # Remove the dock
    dock.free() # Erase the control from the memory
```

While the dock position is chosen when adding it, the user is free to move it and save the layout with the dock in any position.
Checking the results

Now it is the moment to check the results of your work. Open the Project Settings and click on the Plugins tab. Your plugin should be the only on the list. If it is not showing, click on the Update button at the top right corner.

At the Status column, you can see that the plugin is inactive. So you just need to click on the status to select Active. The dock should be immediately visible, even before you close the settings window. And now, lo and behold, you have a custom dock! In just a bit of coding and a simple scene.
7.1.4 Going beyond

Now that you learned how to make basic plugins, you can extend the editor in many nice ways. Many functions can be added to editor on the fly with GDScript, it is a powerful way to create special editors without having to delve into C++ modules.

You can make your own plugins to help you and also share them in Godot’s Asset Library so many people can benefit of your work.
8.1 Introduction to the buildsystem

8.1.1 Scons

Godot uses Scons to build. We love it, we are not changing it for anything else. We are not even sure other build systems are up to the task of building Godot. We constantly get requests to move the build system to CMake, or Visual Studio, but this is not going to happen. There are many reasons why we have chosen SCons over other alternatives and are listed as follows:

- Godot can be compiled for a dozen different platforms. All PC platforms, all mobile platforms, many consoles, and many web-based platforms (such as HTML5 and Chrome PNACL).

- Developers often need to compile for several of the platforms at the same time, or even different targets of the same platform. They can’t afford reconfiguring and rebuilding the project each time. SCons can do this with no sweat, without breaking the builds.

- SCons will never break a build no matter how many changes, configurations, additions, removals etc. You have more chances to die struck by lightning than needing to clean and rebuild in SCons.

- Godot build process is not simple. Several files are generated by code (binders), others are parsed (shaders), and others need to offer customization (plugins). This requires complex logic which is easier to write in an actual programming language (like Python) rather than using a mostly macro-based language only meant for building.

- Godot build process makes heavy use of cross compiling tools. Each platform has a specific detection process, and all these must be handled as specific cases with special code written for each.

So, please get at least a little familiar with it if you are planning to build Godot yourself.

8.1.2 Platform selection

Godot’s build system will begin by detecting the platforms it can build for. If not detected, the platform will simply not appear on the list of available platforms. The build requirements for each platform are described in the rest of this tutorial section.

Scons is invoked by just calling `scons`.

However, this will do nothing except list the available platforms, for example:

```bash
user@host:~/godot$ scons
scons: Reading SConscript files ...
No valid target platform selected.
The following were detected:
```
To build for a platform (for example, x11), run with the `platform=` (or just `p=` to make it short) argument:

```
user@host:~/godot$ scons platform=x11
```

This will start the build process, which will take a while. If you want scons to build faster, use the `-j <cores>` parameter to specify how many cores will be used for the build. Or just leave it using one core, so you can use your computer for something else :)

Example for using 4 cores:

```
user@host:~/godot$ scons platform=x11 -j 4
```

### 8.1.3 Resulting binary

The resulting binaries will be placed in the bin/ subdirectory, generally with this naming convention:

```
godot.<platform>.[opt].[tools/debug].<architecture>[extension]
```

For the previous build attempt the result would look like this:

```
user@host:~/godot$ ls bin
bin/godot.x11.tools.64
```

This means that the binary is for X11, is not optimized, has tools (the whole editor) compiled in, and is meant for 64 bits.

A Windows binary with the same configuration will look like this.

```
C:\\GODOT> DIR BIN/
godot.windows.tools.64.exe
```

Just copy that binary to wherever you like, as it self-contains the project manager, editor and all means to execute the game. However, it lacks the data to export it to the different platforms. For that the export templates are needed (which can be either downloaded from godotengine.org <http://godotengine.org>, or you can build them yourself).

Aside from that, there are a few standard options that can be set in all build targets, and will be explained as follows.

### 8.1.4 Tools

Tools are enabled by default in al PC targets (Linux, Windows, OSX), disabled for everything else. Disabling tools produces a binary that can run projects but that does not include the editor or the project manager.
8.1.5 Target

Target controls optimization and debug flags. Each mode means:

- **debug**: Build with C++ debugging symbols, runtime checks (performs checks and reports error) and none to little optimization.
- **release_debug**: Build without C++ debugging symbols and optimization, but keep the runtime checks (performs checks and reports errors). Official binaries use this configuration.
- **release**: Build without symbols, with optimization and with little to no runtime checks. This target can’t be used together with tools=yes, as the tools require some debug functionality and run-time checks to run.

This flag appends ”.debug” suffix (for debug), or ”.tools” (for debug with tools enabled). When optimization is enabled (release) it appends the ”.opt” suffix.

8.1.6 Bits

Bits is meant to control the CPU or OS version intended to run the binaries. It works mostly on desktop platforms and ignored everywhere else.

- **32**: Build binaries for 32 bits platform.
- **64**: Build binaries for 64 bits platform.
- **default**: Built whatever the build system feels is best. On Linux this depends on the host platform (if not cross compiling), while on Windows and Mac it defaults to produce 32 bits binaries unless 64 bits is specified.

This flag appends ”.32” or ”.64” suffixes to resulting binaries when relevant.

8.1.7 Export templates

Official export templates are downloaded from the Godot Engine site: godotengine.org <http://godotengine.org>. However, you might want to build them yourself (in case you want newer ones, you are using custom modules, or simply don’t trust your own shadow).

If you download the official export templates package and unzip it, you will notice that most are just optimized binaries or packages for each platform:

android_debug.apk
android_release.apk
javascript_debug.zip
javascript_release.zip
linux_server_32
linux_server_64
linux_x11_32_debug
linux_x11_32_release
linux_x11_64_debug
linux_x11_64_release
To create those yourself, just follow the instructions detailed for each platform in this same tutorial section. Each platform explains how to create its own template.

If you are working for multiple platforms, OSX is definitely the best host platform for cross compilation, since you can cross-compile for almost every target (except for winrt). Linux and Windows come in second place, but Linux has the advantage of being the easier platform to set this up.

### 8.2 Compiling for Windows

#### 8.2.1 Requirements

For compiling under Windows, the following is required:

- Visual C++, Visual Studio Community (recommended), at least the 2013 version (12.0) up to 2015 (14.0). Make sure you read Installing Visual Studio caveats below or you will have to run/download the installer again.
- Python 2.7+ (3.0 is untested as of now). Using the 32-bits installer is recommended.
- Pywin32 Python Extension for parallel builds (which increase the build speed by a great factor).
- SCons build system.

#### 8.2.2 Setting up SCons

Python adds the interpreter (python.exe) to the path. It usually installs in C:\Python (or C:\Python[Version]). SCons installs inside the Python install (typically in the Scripts folder) and provides a batch file called “scons.bat”. The location of this file can be added to the path or it can simply be copied to C:\Python together with the interpreter executable.

To check whether you have installed Python and SCons correctly, you can type python --version and scons --version into the standard Windows Command Prompt (cmd.exe).

If commands above do not work, make sure you add Python to your PATH environment variable after installing it, and check again.

#### 8.2.3 Setting up Pywin32

Pywin32 is required for -j (parallel) builds for multiple cores (for a 32 bit Python version). If SCons is issuing a warning about Pywin32 after parsing SConstruct build instructions, when beginning to build, you need to install it properly from the correct installer executable for your python version located at Sourceforge.

For example, if you installed Python 2.7 32 bit version, you would want to install the latest version of Pywin32 (as of writing Build 220) that is built for the mentioned version of Python... That executable installer would be named “pywin32-220.win32-py2.7.exe”.

Amd64 version of Pywin32 is for a 64 bit version of Python “pywin32-220.win-amd64-py2.7.exe”. Change the “py” number to install for your version of python (check via python --version mentioned above).
8.2.4 Installing Visual Studio caveats

If installing VS 2015, make sure to run **Custom** installation, not **Typical** and select C++ as language there (and any other things you might need). The installer does not install C++ by default. C++ was the **only language made optional** in VS2015.

If you have already made the mistake of installing a **Typical**, installation, rerun the executable installer you downloaded from internet, it will give you a **Modify** Button option. Running the install from Add/Remove programs will only give you the “Repair” option, which will do nothing for your problem.

If you’re using Express, make sure you get/have a version that can compile for **C++, Desktop**.

8.2.5 Downloading Godot’s source

Godot’s source is hosted on GitHub. Downloading it (cloning) via Git is recommended.

The tutorial will presume from now on that you placed the source into **C:\godot**.

8.2.6 Compiling

SCons will not be able out of the box to compile from the standard Windows “Command Prompt” (cmd.exe) because SCons and Visual C++ compiler will not be able to locate environment variables and executables they need for compilation.

Therefore, you need to start a Visual Studio command prompt. It sets up environment variables needed by SCons to locate the compiler. It should be called similar to one of the below names (for your respective version of Visual Studio):

- “Developer Command Prompt for VS2013”
- “VS2013 x64 Native Tools Command Prompt”
- “VS2013 x86 Native Tools Command Prompt”
- “VS2013 x64 Cross Tools Command Prompt”
- “VS2013 x86 Cross Tools Command Prompt”

You should be able to find at least the Developer Command Prompt for your version of Visual Studio in your start menu.

However Visual Studio sometimes seems to not install some of the above shortcuts, except the Developer Console at these locations that are automatically searched by the start menu search option:

```plaintext
Win 7:
C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Visual Studio 2015\Visual Studio →Tools
C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Visual Studio 2013\Visual Studio →Tools
```

If you found the Developer Console, it will do for now to create a 32 bit version of Godot, but if you want the 64 bit version, you might need to setup the prompts manually for easy access.

If you don’t see some of the shortcuts, “How the prompts actually work” section below will explain how to setup these prompts if you need them.
About the Developer/Tools Command Prompts and the Visual C++ compiler

There is a few things you need to know about these consoles and the Visual C++ compiler.

Your Visual Studio installation will ship with several Visual C++ compilers, them being more or less identical, however each cl.exe (Visual C++ compiler) will compile Godot for a different architecture (32 or 64 bit, ARM compiler is not supported).

The **Developer Command Prompt** will build a 32 bit version of Godot by using the 32 bit Visual C++ compiler.

**Native Tools** Prompts (mentioned above) are used when you want the 32bit cl.exe to compile a 32 bit executable (x86 Native Tools Command Prompt). For the 64 bit cl.exe, it will compile a 64 bit executable (x64 Native Tools Command Prompt).

The **Cross Tools** are used when your Windows is using one architecture (32 bit, for example) and you need to compile to a different architecture (64 bit). As you might be familiar, 32 bit Windows can not run 64 bit executables, but you still might need to compile for them.

For example:

- “VS2013 x64 Cross Tools Command Prompt” will use a 32 bit cl.exe that will compile a 64 bit application.
- “VS2013 x86 Cross Tools Command Prompt” will use a 64 bit cl.exe that will compile a 32 bit application. This one is useful if you are running a 32 bit Windows.

On a 64 bit Windows, you can run any of above prompts and compilers (cl.exe executables) because 64 bit windows can run any 32 bit application. 32 bit Windows can not run 64 bit executables, so the Visual Studio installer will not even install shortcuts for some of these prompts.

Note that you need to choose the **Developer Console** or the correct **Tools Prompt** to build Godot for the correct architecture. Use only Native Prompts if you are not sure yet what exactly Cross Compile Prompts do.

Running SCons

Once inside the **Developer Console/Tools Console Prompt**, go to the root directory of the engine source code and type:

```
C:\godot> scons platform=windows
```

Tip: if you installed “Pywin32 Python Extension” you can append the -j command to instruct SCons to run parallel builds like this:

```
C:\godot> scons -j6 platform=windows
```

In general, it is OK to have at least as many threads compiling Godot as you have cores in your CPU, if not one or two more. I use -j6 (six threads) for my 4 core CPU, your mileage may vary. Feel free to add -j option to any SCons command you see below if you setup the “Pywin32 Python Extension”.

If all goes well, the resulting binary executable will be placed in `C:\godot\bin\` with the name of `godot.windows.tools.32.exe` or `godot.windows.tools.64.exe`. SCons will automatically detect what compiler architecture the environment (the prompt) is setup for and will build a corresponding executable.

This executable file contains the whole engine and runs without any dependencies. Executing it will bring up the project manager.

How the prompts actually work

The Visual Studio command prompts are just shortcuts that call the standard Command Prompt and have it run a batch file before giving you control. The batch file itself is called `vcvarsall.bat` and it sets up environment variables,
including the PATH variable, so that the correct version of the compiler can be run. The Developer Command Prompt calls a different file called `VsDevCmd.bat` but none of the other tools that this batch file enables are needed by Godot/SCons.

Since you are probably using VS2013 or VS2015, if you need to recreate them manually, use the below folders, or place them on the desktop/taskbar:

```plaintext
C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Visual Studio 2015\Visual Studio Tools
C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Visual Studio 2013\Visual Studio Tools
```

Start the creation of the shortcut by pressing the right mouse button/New/Shortcut in an empty place in your desired location.

Then copy one of these commands below for the corresponding tool you need into the “Path” and “Name” sections of the shortcut creation wizard, and fix the path to the batch file if needed.

- VS2013 is in the “Microsoft Visual Studio 12.0” folder.
- VS2015 is in the “Microsoft Visual Studio 14.0” folder.
- etc.

```plaintext
Name: Developer Command Prompt for VS2013
Path: %comspec% /k ""C:\Program Files (x86)\Microsoft Visual Studio 12.0\Common7\Tools\VsDevCmd.bat"

Name: VS2013 x64 Cross Tools Command Prompt
Path: %comspec% /k ""C:\Program Files (x86)\Microsoft Visual Studio 12.0\VC\vcvarsall.bat"" x86_amd64

Name: VS2013 x64 Native Tools Command Prompt
Path: %comspec% /k ""C:\Program Files (x86)\Microsoft Visual Studio 12.0\VC\vcvarsall.bat"" amd64

Name: VS2013 x86 Native Tools Command Prompt
Path: %comspec% /k ""C:\Program Files (x86)\Microsoft Visual Studio 12.0\VC\vcvarsall.bat"" x86

Name: VS2013 x86 Cross Tools Command Prompt
Path: %comspec% /k ""C:\Program Files (x86)\Microsoft Visual Studio 12.0\VC\vcvarsall.bat"" amd64_x86

Name: VS2013 x86 (32bit) cl.exe
Path: C:\Program Files (x86)\Microsoft Visual Studio 12.0\VC\bin\cl.exe

Name: VS2013 x64 (32bit) cl.exe for crosscompiling to 64bit.
Path: C:\Program Files (x86)\Microsoft Visual Studio 12.0\VC\bin\x86_amd64\cl.exe

Name: VS2013 x64 (64bit) cl.exe
Path: C:\Program Files (x86)\Microsoft Visual Studio 12.0\VC\bin\amd64\cl.exe
```

After you create the shortcut, in the shortcut’s properties, that you can access by right clicking with your mouse on the shortcut itself, you can choose the starting directory of the command prompt (“Start in” field).

Some of these shortcuts (namely the 64bit compilers) seem to not be available in the Express edition of Visual Studio or Visual C++. Before recreating the commands, make sure that `cl.exe` executables are present in one of these locations, they are the actual compilers for the architecture you want to build from the command prompt.

```plaintext
x86 (32bit) cl.exe
C:\Program Files (x86)\Microsoft Visual Studio 12.0\VC\bin\cl.exe

x86 (32bit) cl.exe for crosscompiling to 64bit.
C:\Program Files (x86)\Microsoft Visual Studio 12.0\VC\bin\x86_amd64\cl.exe

x64 (64bit) cl.exe
C:\Program Files (x86)\Microsoft Visual Studio 12.0\VC\bin\amd64\cl.exe
```
x64 (64bit) cl.exe for crosscompiling to 32bit.
C:\Program Files (x86)\Microsoft Visual Studio 12.0\VC\bin\amd64_x86\cl.exe

In case you are wondering what these prompt shortcuts do, they call the standard cmd.exe with \k option and have it run a batch file...

%comspec% - path to cmd.exe
\k - keep alive option of the command prompt
remainder - command to run via cmd.exe
cmd.exe \k (keep cmd.exe alive after commands behind this option run) "runme.bat" ˓→with_this_option

How to run an automated build of Godot

If you need to just run the compilation process via a batch file or directly in the vanilla Windows Command Prompt you need to do the following command:

"C:\Program Files (x86)\Microsoft Visual Studio 12.0\VC\vcvarsall.bat" x86

with one of the following parameters:
- x86 (32 bit cl.exe to compile for the 32 bit architecture)
- amd64 (64 bit cl.exe to compile for the 64 bit architecture)
- x86_amd64 (32 bit cl.exe to compile for the 64 bit architecture)
- amd64_x86 (64 bit cl.exe to compile for the 32 bit architecture)

and after that one, you can run SCons:
scons platform=windows

or you can do them together:

32 bit Godot
"C:\Program Files (x86)\Microsoft Visual Studio 12.0\VC\vcvarsall.bat" x86 && scons ˓→platform=windows

64 bit Godot
"C:\Program Files (x86)\Microsoft Visual Studio 12.0\VC\vcvarsall.bat" amd64 && scons ˓→platform=windows

8.2.7 Development in Visual Studio or other IDEs

For most projects, using only scripting is enough but when development in C++ is needed, for creating modules or extending the engine, working with an IDE is usually desirable.

You can create a Visual Studio solution via SCons by running SCons with the vsproj=yes parameter, like this:
scons p=windows vsproj=yes

You will be able to open Godot’s source in a Visual Studio solution now, and able to build Godot via the Visual Studio Build button. However, make sure that you have installed Pywin so that parallel (-j) builds work properly.
If you need to edit the compilation commands, they are located in “Godot” project settings, NMAKE sheet. SCons is called at the very end of the commands. If you make a mistake, copy the command from one of the other build configurations (debug, release_debug, release) or architectures (Win32/x64). They are equivalent.

### 8.2.8 Cross-compiling for Windows from other operating systems

If you are a Linux or Mac user, you need to install MinGW-w64, which typically comes in 32-bit and 64-bit variants. The package names may differ based on your distro, here are some known ones:

<table>
<thead>
<tr>
<th>Arch</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arch</td>
<td>pacman -S scons mingw-w64-gcc</td>
</tr>
<tr>
<td>Debian / Ubuntu</td>
<td>apt-get install scons mingw-w64</td>
</tr>
<tr>
<td>Fedora</td>
<td>dnf install scons mingw32-gcc-c++ mingw64-gcc-c++</td>
</tr>
<tr>
<td>macOS</td>
<td>brew install scons mingw-w64</td>
</tr>
<tr>
<td>Mageia</td>
<td>urpmi scons mingw32-gcc-c++ mingw64-gcc-c++</td>
</tr>
</tbody>
</table>

Before allowing you to attempt the compilation, SCons will check for the following binaries in your `$PATH`:

<table>
<thead>
<tr>
<th></th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>i686-w64-mingw32-gcc</td>
<td></td>
</tr>
<tr>
<td>x86_64-w64-mingw32-gcc</td>
<td></td>
</tr>
</tbody>
</table>

If the binaries are not located in the `$PATH` (e.g. `/usr/bin`), you can define the following environment variables to give a hint to the buildsystem:

```
export MINGW32_PREFIX="/path/to/i686-w64-mingw32-"
export MINGW64_PREFIX="/path/to/x86_64-w64-mingw32-"
```

To make sure you are doing things correctly, executing the following in the shell should result in a working compiler (the version output may differ based on your system):

```
user@host:~$ ${MINGW32_PREFIX}/gcc --version
```

8.2.9 Creating Windows export templates

Windows export templates are created by compiling Godot as release, with the following flags:

- (using Mingw32 command prompt, using the bits parameter)

```
C:\godot> scons platform=windows tools=no target=release bits=32
C:\godot> scons platform=windows tools=no target=release_debug bits=32
```

- (using Mingw-w64 command prompt, using the bits parameter)

```
C:\godot> scons platform=windows tools=no target=release bits=64
C:\godot> scons platform=windows tools=no target=release_debug bits=64
```

- (using the Visual Studio command prompts for the correct architecture, notice the lack of bits parameter)
If you plan on replacing the standard templates, copy these to:

```
C:\\Users\\YOURUSER\\AppData\\Roaming\\Godot\\Templates
```

With the following names:

```
windows_32_debug.exe
windows_32_release.exe
windows_64_debug.exe
windows_64_release.exe
```

However, if you are writing your custom modules or custom C++ code, you might instead want to configure your binaries as custom export templates here:

![Project Export Settings](image)

You don’t even need to copy them, you can just reference the resulting files in the bin\ directory of your Godot source folder, so the next time you build you automatically have the custom templates referenced.
8.3 Compiling for X11 (Linux, *BSD)

8.3.1 Requirements

For compiling under Linux or other Unix variants, the following is required:

• GCC < 6 or Clang (warning: see note below regarding GCC 6).
• Python 2.7+ (3.0 is untested as of now)
• SCons build system
• pkg-config (used to detect the dependencies below)
• X11, Xcursor, Xinerama and XRandR development libraries
• MesaGL development libraries
• ALSA development libraries
• PulseAudio development libraries (for sound support)
• Freetype (for the editor)
• OpenSSL (for HTTPS and TLS)
• libudev-dev (optional, build with udev=yes)

Known issues with GCC 6

There are known optimisation issues when using GCC 6 (both to compile for X11 and to cross-compile for Windows with MinGW). Until those issues are fixed in Godot’s source, or the compiler is made more forgiving again, it is advised not to use GCC 6 to compile Godot as release builds will trigger crashes.

If your distribution provides GCC 6 by default (e.g. Arch or Ubuntu 16.10), you may have to install an older version or Clang (both should be provided in the repositories). You can then force the use of another compiler version via the CC and CXX scons arguments (e.g. scons p=x11 CC=gcc-5 CXX=g++-5 if your distribution provides those binaries in its PATH). You can also use llvm=yes instead to force the usage of Clang over GCC.
## Distro-specific oneliners

<table>
<thead>
<tr>
<th>Arch</th>
<th>Arch-specific commands</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arch</td>
<td>pacman -S scons libxcursor libxinerama libxrandr mesa glu alsa-lib pulseaudio freetype2</td>
</tr>
<tr>
<td>Debian / Ubuntu</td>
<td>sudo apt-get install build-essential scons pkg-config libgl1-mesa-dev libglu-dev libasdound2-dev libxrandr-dev</td>
</tr>
<tr>
<td>Fedora</td>
<td>sudo dnf install scons pkgconfig libX11-devel libX11-devel libGL1-devel alsa-devel pulseaudio-devel mesa-mesa-libGLU-devel</td>
</tr>
<tr>
<td>FreeBSD</td>
<td>sudo pkg install scons pkgconfig xorg-libraries libx11 libfreetype2 openssl</td>
</tr>
<tr>
<td>Gentoo</td>
<td>emerge -an dev-util/scons x11-libs/libX11 x11-libs/media-libs/glu media-libs/alsa libmedia-sound</td>
</tr>
<tr>
<td>Mageia</td>
<td>urpmi scons pkgconfig &quot;pkgconfig(alsa)&quot; &quot;pkgconfig(freetype)&quot; &quot;pkgconfig(udev)&quot; &quot;pkgconfig(xrandr)&quot;</td>
</tr>
<tr>
<td>OpenBSD</td>
<td>pkg_add python scons png llvm</td>
</tr>
<tr>
<td>openSUSE</td>
<td>sudo zypper install scons pkgconfig libX11-devel libX11-devel Mesa-libGL-devel alsa-devel libpulse-devel libGLU1</td>
</tr>
<tr>
<td>Solus</td>
<td>sudo eopkg install -c system.devel scons libxcursor mesalib-devel libglu alsa-lib pulseaudio freetype2</td>
</tr>
</tbody>
</table>

### 8.3.2 Compiling

Start a terminal, go to the root dir of the engine source code and type:

```
user@host:~/godot$ scons platform=x11
```

If all goes well, the resulting binary executable will be placed in the “bin” subdirectory. This executable file contains the whole engine and runs without any dependencies. Executing it will bring up the project manager.

**Note:** If you wish to compile using Clang rather than GCC, use this command:

```
user@host:~/godot$ scons platform=x11 use_llvm=yes
```

Using Clang appears to be a requirement for OpenBSD, otherwise fonts would not build.
### 8.3.3 Building export templates

To build X11 (Linux, *BSD) export templates, run the build system with the following parameters:

- (32 bits)

```bash
user@host:~/godot$ scons platform=x11 tools=no target=release bits=32
user@host:~/godot$ scons platform=x11 tools=no target=release_debug bits=32
```

- (64 bits)

```bash
user@host:~/godot$ scons platform=x11 tools=no target=release bits=64
user@host:~/godot$ scons platform=x11 tools=no target=release_debug bits=64
```

Note that cross compiling for the opposite bits (64/32) as your host platform is not always straight-forward and might need a chroot environment.

To create standard export templates, the resulting files must be copied to:

```
/home/youruser/.godot/templates
```

and named like this (even for *BSD which is seen as “Linux X11” by Godot):

```
linux_x11_32_debug
linux_x11_32_release
linux_x11_64_debug
linux_x11_64_release
```

However, if you are writing your custom modules or custom C++ code, you might instead want to configure your binaries as custom export templates here:
You don’t even need to copy them, you can just reference the resulting files in the bin/ directory of your Godot source folder, so the next time you build you automatically have the custom templates referenced.

8.4 Compiling for OSX

8.4.1 Requirements

For compiling under Linux or other Unix variants, the following is required:

- Python 2.7+ (3.0 is untested as of now)
- SCons build system
- Xcode (or the more lightweight Command Line Tools for Xcode)

8.4.2 Compiling

Start a terminal, go to the root dir of the engine source code and type:

```
user@host:/$godot$ scons platform=osx
```
If all goes well, the resulting binary executable will be placed in the “bin” subdirectory. This executable file contains the whole engine and runs without any dependencies. Executing it will bring up the project manager.

To create an .app like in the official builds, you need to use the template located in `misc/dist/osx_tools.app`. Typically, for a “fat” binary that supports both 32-bit and 64-bit architectures, and with an optimised binary built with `scons p=osx target=release_debug`:

```
user@host:~/godot$ cp -r misc/dist/osx_tools.app ./Godot.app
user@host:~/godot$ mkdir -p Godot.app/Contents/MacOS
user@host:~/godot$ cp bin/godot.osx.opt.tools.fat Godot.app/Contents/MacOS/Godot
user@host:~/godot$ chmod +x Godot.app/Contents/MacOS/Godot
```

### 8.4.3 Cross-compiling

It is possible to compile for OSX in a Linux environment (and maybe also in Windows with Cygwin). For that you will need OSXCross to be able to use OSX as target. First, follow the instructions to install it:

- Clone the OSXCross repository <https://github.com/tpoechtrager/osxcross> somewhere on your machine (or download a zip file and extract it somewhere), e.g.:

```
user@host:~$ git clone https://github.com/tpoechtrager/osxcross.git /home/myuser/sources/osxcross
```

1. Follow the instructions to package the SDK: https://github.com/tpoechtrager/osxcross#packaging-the-sdk
2. Follow the instructions to install OSXCross: https://github.com/tpoechtrager/osxcross#installation

After that, you will need to define the `OSXCROSS_ROOT` as the path to the OSXCross installation (the same place where you cloned the repository/extracted the zip), e.g.:

```
user@host:~$ export OSXCROSS_ROOT=/home/myuser/sources/osxcross
```

Now you can compile with SCons like you normally would:

```
user@host:~/godot$ scons platform=osx
```

If you have an OSXCross SDK version different from the one expected by the SCons buildsystem, you can specify a custom one with the `osxcross_sdk` argument:

```
user@host:~/godot$ scons platform=osx osxcross_sdk=darwin15
```

### 8.5 Compiling for Android

#### 8.5.1 Note

In most cases, using the built-in deployer and export templates is good enough. Compiling the Android APK manually is mostly useful for custom builds or custom packages for the deployer.

Also, you still need to follow the steps mentioned in the *Exporting for Android* tutorial before attempting to build a custom export template.
8.5.2 Requirements

For compiling under Windows, Linux or OSX, the following is required:

- Python 2.7+ or Python 3.5+
- SCons build system
- Android SDK (command-line tools are sufficient)
  - Required SDK components will be automatically installed by Gradle (except the NDK)
- Android NDK r17 or later
- Gradle (will be downloaded and installed automatically if missing)
- JDK 8 (either OpenJDK or Oracle JDK)
  - JDK 9 or later are not currently supported
  - You can download a build from ojdkbuild

See also:

For a general overview of SCons usage for Godot, see Introduction to the buildsystem.

8.5.3 Setting up the buildsystem

Set the environment variable ANDROID_HOME to point to the Android SDK. If you downloaded the Android command-line tools, this would be the folder where you extracted the contents of the ZIP archive. Later on, gradlew will install necessary SDK components in this folder. However, you need to accept the SDK component licenses before they can be downloaded by Gradle. This can be done by running the following command from the root of the SDK directory, then answering all the prompts with y:

```
tools/bin/sdkmanager --licenses
```

Set the environment variable ANDROID_NDK_ROOT to point to the Android NDK.

To set those environment variables on Windows, press Windows + R, type “control system”, then click on Advanced system settings in the left pane, then click on Environment variables on the window that appears.

To set those environment variables on Linux or macOS, use export ANDROID_HOME=/path/to/android-sdk and export ANDROID_NDK_ROOT=/path/to/android-ndk where /path/to/android-sdk and /path/to/android-ndk point to the root of the SDK and NDK directories.

Toolchain

We usually try to keep the Godot Android build code up to date, but Google changes their toolchain versions very often, so if compilation fails due to wrong toolchain version, go to your NDK directory and check the current number, then set the following environment variable:

```
NDK_TARGET (by default set to "arm-linux-androideabi-4.9")
```
8.5.4 Building the export templates

Godot needs two export templates for Android: the optimized “release” template (android_release.apk) and the debug template (android_debug.apk). As Google will require all APKs to include ARMv8 (64-bit) libraries starting from August 2019, the commands below will build an APK containing both ARMv7 and ARMv8 libraries.

Compiling the standard export templates is done by calling SCons with the following arguments:

- **Release template** (used when exporting with Debugging Enabled unchecked)

  ```
  scons platform=android target=release android_arch=armv7
  scons platform=android target=release android_arch=arm64v8
  cd platform/android/java
  # On Windows
  .\gradlew build
  # On Linux and macOS
  ./gradlew build
  ```

  The resulting APK will be located at `bin/android_release.apk`.

- **Debug template** (used when exporting with Debugging Enabled checked)

  ```
  scons platform=android target=release_debug android_arch=armv7
  scons platform=android target=release_debug android_arch=arm64v8
  cd platform/android/java
  # On Windows
  .\gradlew build
  # On Linux and macOS
  ./gradlew build
  ```

  The resulting APK will be located at `bin/android_debug.apk`.

Adding support for x86 devices

If you also want to include support for x86 devices, run the SCons command a third time with the `android_arch=x86` argument before building the APK with Gradle. For example, for the release template:

```
# On Windows
 .\gradlew build
 # On Linux and macOS
 ./gradlew build
```

This will create a fat binary that works on all platforms. The final APK size of exported projects will depend on the platforms you choose to support when exporting; in other words, unused platforms will be removed from the APK.

8.5.5 Using the export templates

Godot needs release and debug APKs that were compiled against the same version/commit as the editor. If you are using official binaries for the editor, make sure to install the matching export templates, or build your own from the same version.

When exporting your game, Godot opens the APK, changes a few things inside, adds your file and spits it back. It’s really handy! (and required some reverse engineering of the format).
Installing the templates

The newly-compiled templates (android_debug.apk and android_release.apk) must be copied to Godot's templates folder with their respective names. The templates folder can be located in:

- **Windows:** `%APPDATA%\Godot\templates\<version>\`
- **Linux:** `$HOME/.local/share/godot/templates/<version>/`
- **macOS:** `$HOME/Library/Application Support/Godot/templates/<version>/`

<version> is of the form major.minor[.patch].status using values from version.py in your Godot source repository (e.g. 3.0.5.stable or 3.1.dev). You also need to write this same version string to a version.txt file located next to your export templates.

However, if you are writing your custom modules or custom C++ code, you might instead want to configure your APKs as custom export templates here:

![Project Export Settings](image)

You don’t even need to copy them, you can just reference the resulting file in the bin\ directory of your Godot source folder, so that the next time you build you will automatically have the custom templates referenced.

### 8.5.6 Troubleshooting
Application not installed

Android might complain the application is not correctly installed. If so:

- Check that the debug keystore is properly generated.
- Check that the jarsigner executable is from JDK 8.

If it still fails, open a command line and run `logcat`:

```
adb logcat
```

Then check the output while the application is installed; the error message should be presented there. Seek assistance if you can’t figure it out.

Application exits immediately

If the application runs but exits immediately, this might be due to one of the following reasons:

- Make sure to use export templates that match your editor version; if you use a new Godot version, you have to update the templates too.
- `libgodot_android.so` is not in `libs/<android_arch>/` where `<android_arch>` is the device’s architecture.
- The device’s architecture does not match the exported one(s). Make sure your templates were built for that device’s architecture, and that the export settings included support for that architecture.

In any case, `adb logcat` should also show the cause of the error.

8.6 Compiling for iOS

8.6.1 Requirements

- SCons (you can get it from macports, you should be able to run `scons` in a terminal when installed)
- Xcode with the iOS SDK and the command line tools.

8.6.2 Compiling

Open a Terminal, go to the root dir of the engine source code and type:

```
$ scons p=iphone target=debug
```

for a debug build, or:

```
$ scons p=iphone target=release
```

for a release build (check `platform/iphone/detect.py` for the compiler flags used for each configuration). Alternatively, you can run

```
$ scons p=iphone arch=x86 target=debug
```
for a Simulator executable. Additionally since some time Apple requires 64 bit version of application binary when you are uploading to iStore. The best way to provide these is to create a bundle in which there are both 32bit and 64 binaries, so every device will be able to run the game. It can be done in three steps, first compile 32 bit version, then compile 64 bit version and then use lipo to bundle them into one fat binary, all those steps can be performed with following commands:

```
$ scons p=iphone tools=no bits=32 target=release arch=arm
$ scons p=iphone tools=no bits=64 target=release arch=arm64
$ lipo -create bin/godot.iphone.opt.32 bin/godot.iphone.opt.64 -output bin/godot.→iphone.opt.universal
```

8.6.3 Run

To run on a device or simulator, follow these instructions: Exporting for iOS.

Replace or add your executable to the Xcode project, and change the “executable name” property on Info.plist accordingly if you use an alternative build.

8.7 Cross-compiling for iOS on Linux

The procedure for this is somewhat complex and requires a lot of steps, but once you have the environment properly configured it will be easy to compile Godot for iOS anytime you want.

8.7.1 Disclaimer

While it is possible to compile for iOS on a Linux environment, Apple is very restrictive about the tools to be used (specially hardware-wise), allowing pretty much only their products to be used for development. So this is not official. However, a statement from Apple in 2010 says they relaxed some of the App Store review guidelines to allow any tool to be used, as long as the resulting binary does not download any code, which means it should be OK to use the procedure described here and cross-compiling the binary.

8.7.2 Requirements

- XCode with the iOS SDK (a dmg image)
- Clang &gt;= 3.5 for your development machine installed and in the PATH. It has to be version &gt;= 3.5 to target arm64 architecture.
- Fuse for mounting and umounting the dmg image.
- darling-dmg, which needs to be built from source. The procedure for that is explained below.
  - For building darling-dmg, you’ll need the development packages of the following libraries: fuse, icu, openssl, zlib, bzip2.
- cctools-port for the needed build tools. The procedure for building is quite peculiar and is described below.
  - This also has some extra dependencies: automake, autogen, libtool.
8.7.3 Configuring the environment

darling-dmg

Clone the repository on your machine:

```
$ git clone https://github.com/darlinghq/darling-dmg.git
```

Build it:

```
$ cd darling-dmg
$ mkdir build
$ cd build
$ cmake .. -DCMAKE_BUILD_TYPE=Release
$ make -j 4  # The number is the amount of cores your processor has, for faster build
$ cd ../
```

Preparing the SDK

Mount the XCode image:

```
$ mkdir xcode
$ ./darling-dmg/build/darling-dmg /path/to/Xcode_7.1.1.dmg xcode
[...]
Everything looks OK, disk mounted
```

Extract the iOS SDK:

```
$ mkdir -p iPhoneSDK/iPhoneOS9.1.sdk
$ cp -r xcode/Xcode.app/Contents/Developer/Platforms/iPhoneOS.platform/Developer/SDKs/ iPhoneOS.sdk/* iPhoneSDK/iPhoneOS9.1.sdk
$ cp -r xcode/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/ include/c++/* iPhoneSDK/iPhoneOS9.1.sdk/usr/include/c++
$ fusermount -u xcode  # unmount the image
```

Pack the SDK:

```
$ cd iPhoneSDK
$ tar -cf - * | xz -9 -c - > iPhoneOS9.1.sdk.tar.xz
```

Toolchain

Build cctools:

```
$ git clone https://github.com/tpoechtrager/cctools-port.git
$ cd cctools-port/usage_examples/ios_toolchain
$ ./build.sh /path/iPhoneOS9.1.sdk.tar.xz arm64
```

Copy the tools to a nicer place. Note that the SCons scripts for building will look under `usr/bin` inside the directory you provide for the toolchain binaries, so you must copy to such subdirectory, akin to the following commands:

```
$ mkdir -p /home/user/iostoolchain/usr
$ cp -r target/bin /home/user/iostoolchain/usr/
```

Now you should have the iOS toolchain binaries in `/home/user/iostoolchain/usr/bin`. 

8.7. Cross-compiling for iOS on Linux
8.7.4 Compiling Godot for iPhone

Once you’ve done the above steps, you should keep two things in your environment: the built toolchain and the iPhoneOS SDK directory. Those can stay anywhere you want since you have to provide their paths to the SCons build command.

For the iPhone platform to be detected, you need the `OSXCROSS_IOS` environment variable defined to anything.

```
$ export OSXCROSS_IOS=anything
```

Now you can compile for iPhone using SCons like the standard Godot way, with some additional arguments to provide the correct paths:

```
$ scons -j 4 platform=iphone arch=arm target=release_debug IPHONESDK="/path/to/iphonesdk" IPHONEPATH="/path/to/iostoolchain" ios_triple="arm-apple-darwin11-
$ scons -j 4 platform=iphone arch=arm64 target=release_debug IPHONESDK="/path/to/iphonesdk" IPHONEPATH="/path/to/iostoolchain" ios_triple="arm-apple-darwin11-
```

Producing fat binaries

Apple requires a fat binary with both architectures (armv7 and arm64) in a single file. To do this, use the `arm-apple-darwin11-lipo` executable. The following example assumes you are in the root Godot source directory:

```
$ /path/to/iostoolchain/usr/bin/arm-apple-darwin11-lipo -create bin/godot.iphone.opt.debug.arm bin/godot.iphone.opt.debug.arm64 -output bin/godot.iphone.opt.debug.fat
```

Then you will have an iOS fat binary in `bin/godot.iphone.opt.debug.fat`.

8.8 Compiling for Universal Windows Platform

8.8.1 Requirements

- SCons (see *Compiling for Windows* for more details).
- Visual Studio 2015 Update 2. It may work with earlier versions. See *Compiling for Windows* about the caveats of installing it and the various prompts.
- Windows 10 SDK (can be selected in Visual Studio installation).
- ANGLE source. Use the `ms_master` (default) branch. Keep it in a path without spaces to avoid problems.

8.8.2 Compiling

You need to open a proper Visual Studio prompt for the target architecture you want to build. Check *Compiling for Windows* to see how these prompts work.

There are three target architectures for UWP: x86 (32-bits), x64 (64-bits) and ARM (32-bits). You can run `vcvarsall.bat` with arm as argument to set the environment for this target architecture.

Set the `ANGLE_SRC_PATH` to the directory where you downloaded the ANGLE source code. The build process will also build ANGLE to produce the required DLLs for the selected architecture.

Once you’re set, run the SCons command similarly to the other platforms:
8.8.3 Creating UWP export templates

To export using the editor you need to properly build package the templates. You need all three architectures with debug and release templates to be able to export.

Open the command prompt for one architecture and run SCons twice (once for each target):

```class
C:\godot>scons platform=winrt target=release_debug
C:\godot>scons platform=winrt target=release
```

Repeat for the other architectures.

In the end your bin folder will have the .exe binaries with a name like godot.winrt.opt.debug.32.x86.exe (with variations for each target/arch).

Copy one of these to misc/dist/uwp_template inside the Godot source folder and rename the binary to godot.winrt.exe. From the ANGLE source, under winrt/10/src/Release_%arch% (where %arch% can be Win32, x64 or ARM), get the libEGL.dll and the libGLESv2.dll, putting them along with the executable.

Add the files in the uwp_template folder to a ZIP. Rename the resulting Zip according to the target/architecture of the template:

```class
winrt_x86_debug.zip
winrt_x86_release.zip
winrt_x64_debug.zip
winrt_x64_release.zip
winrt_arm_debug.zip
winrt_arm_release.zip
```

Move those templates to the templates folder in Godot settings path. If you don’t want to replacet the templates, you can set the “Custom Package” property in the export window.

8.8.4 Running UWP apps with Visual Studio

If you want to debug the UWP port or simply run your apps without packaging and signing, you can deploy and launch them using Visual Studio. It might be the easiest way if you are testing on a device such as a Windows Phone or an Xbox One.

Within the ANGLE source folder, open templates and double-click the install.bat script file. This will install the Visual Studio project templates for ANGLE apps.

If you have not built Godot yet, open the winrt/10/src/angle.sln solution from the ANGLE source and build it to Release/Win32 target. You may also need to build it for ARM if you plan to run on a device. You can also use MSBuild if you’re comfortable with the command line.

Create a new Windows App project using the “App for OpenGL ES (Windows Universal)” project template, which can be found under the Visual C++/Windows/Universal category.

This is a base project with the ANGLE dependencies already set up. However, by default it picks the debug version of the DLLs which usually have a very poor performance. So in the “Binaries” filter, click in each of the DLLs there and in the “Properties” window and change the relative path from Debug_Win32 to Release_Win32 (or Release_ARM for devices).

8.8. Compiling for Universal Windows Platform
In the same “Binaries” filter, select “Add > Existing Item” and point to the Godot executable for UWP you have. In the “Properties” window, set “Content” to True so it’s included in the project.

Right-click the Package.appxmanifest file and select “Open With... > XML (Text) Editor”. In the Package/Applications/Application element, replace the Executable attribute from $targetnametoken$.exe to godot.winrt.exe (or whatever your Godot executable is called). Also change the EntryPoint attribute to GodotWinRT.App. This will ensure that the Godot executable is correctly called when the app starts.

Create a folder (not a filter) called game in your Visual Studio project folder and there you can put either a data.pck file or your Godot project files. After that, make sure to include it all with the “Add > Existing Item” command and set their “Content” property to True so they’re copied to the app.

To ease the workflow, you can open the “Solution Properties” and in the “Configuration” section untick the “Build” option for the app. You still have to build it at least once to generate some needed files, you can do so by right-clicking the project (not the solution) in the “Solution Explorer” and selecting “Build”.

Now you can just run the project and your app should open. You can use also the “Start Without Debugging” from the “Debug” menu (Ctrl+F5) to make it launch faster.

### 8.9 Compiling for the Web

#### 8.9.1 Requirements

To compile export templates for the Web, the following is required:

- Emscripten SDK (Install in a path without spaces, i.e. not on “Program Files”)
- Python 2.7+ (3.0 is untested as of now)
- SCons build system

#### 8.9.2 Compiling

Start a terminal and set the environment variable EMSCRIPTEN_ROOT to the installation directory of Emscripten:

```bash
export EMSCRIPTEN_ROOT=~/emsdk/emsdk/emsdk
```

If you are on Windows, start a regular prompt or the Emscripten Command Prompt. Do not use the Developer Command Prompt nor any of the ones that come with Visual Studio. You can set the environment variable in the system settings or in the prompt itself:

```cmd
set EMSCRIPTEN_ROOT=C:\emsdk\emsdk\master
```

Now go to the root directory of the engine source code and instruct SCons to compile for JavaScript. Specify target as either release for a release build or release_debug for a debug build:

```bash
scons platform=javascript tools=no target=release
scons platform=javascript tools=no target=release_debug
```

The engine will now be compiled to JavaScript by Emscripten. If all goes well, the resulting file will be placed in the bin subdirectory. Its name is godot.javascript.opt.asm.js for release or godot.javascript.opt.debug.asm.js for debug. Additionally, two files of the same name but with the extensions .js and .html.mem will be generated.
8.9.3 Building export templates

After compiling, further steps are required to build the template. The actual web export template has the form of a zip file containing at least these 5 files:

1. godot.asm.js — This is the file that was just compiled, but under a different name.
   
   For the release template:
   ```bash
cp bin/godot.javascript.opt.asm.js godot.asm.js
   ```
   
   For the debug template:
   ```bash
   cp bin/godot.javascript.opt.debug.asm.js godot.asm.js
   ```

2. godot.js

3. godot.mem — other files created during compilation, initially with the same name as the .asm.js file, except .asm.js is replaced by .js for godot.js and .html.mem for godot.mem.
   
   For the release template:
   ```bash
   cp bin/godot.javascript.opt.js godot.js
cp bin/godot.javascript.opt.html.mem godot.mem
   ```
   
   For the debug template:
   ```bash
   cp bin/godot.javascript.opt.debug.js godot.js
cp bin/godot.javascript.opt.debug.html.mem godot.mem
   ```

4. godot.html

5. godotfs.js — Both of these files are located within the Godot Engine repository, under tools/dist/html_fs/.
   ```bash
   cp tools/dist/html_fs/godot.html .
cp tools/dist/html_fs/godotfs.js .
   ```

Once these 5 files are assembled, zip them up and your export template is ready to go. The correct name for the template file is javascript_release.zip for the release template:
```bash
zip javascript_release.zip godot.asm.js godot.js godot.mem godotfs.js godot.html
```

And javascript_debug.zip for the debug template:
```bash
zip javascript_debug.zip godot.asm.js godot.js godot.mem godotfs.js godot.html
```

The resulting files must be placed in the templates directory in your Godot user directory:
```bash
mv javascript_release.zip ~/.godot/templates
mv javascript_debug.zip ~/.godot/templates
```

If you are writing custom modules or using custom C++ code, you may want to configure your zip files as custom export templates. This can be done in the export GUI, using the “Custom Package” option. There’s no need to copy the templates in this case — you can simply reference the resulting files in your Godot source folder, so the next time you build, the custom templates will already be referenced.

8.9. Compiling for the Web
8.9.4 Customizing the HTML page

Rather than the default godot.html file from the Godot Engine repository’s tools/dist/html_fs/ directory, it is also possible to use a custom HTML page. This allows drastic customization of the final web presentation.

In the HTML page, the JavaScript object Module is the page’s interface to Emscripten. Check the official documentation for information on how to use it: https://kripken.github.io/emscripten-site/docs/api_reference/module.html

The default HTML page offers an example to start off with, separating the Emscripten interface logic in the JavaScript Module object from the page logic in the Presentation object. Emscripten’s default shell.html file is another example, but does not use Godot’s placeholders, listed below.

When exporting a game, several placeholders in the godot.html file are substituted by values dependent on the export:

<table>
<thead>
<tr>
<th>Placeholder</th>
<th>Substituted by</th>
</tr>
</thead>
<tbody>
<tr>
<td>$GODOT_JS</td>
<td>Name of the compiled Godot Engine JavaScript file</td>
</tr>
<tr>
<td>$GODOT_FS</td>
<td>Name of the filesystem access JavaScript file</td>
</tr>
<tr>
<td>$GODOT_MEM</td>
<td>Name of the memory initialization file</td>
</tr>
<tr>
<td>$GODOT_CANVAS_WIDTH</td>
<td>Integer specifying the initial display width of the game</td>
</tr>
<tr>
<td>$GODOT_CANVAS_HEIGHT</td>
<td>Integer specifying the initial display height of the game</td>
</tr>
<tr>
<td>$GODOT_DEBUG_ENABLED</td>
<td>String true if debugging, false otherwise</td>
</tr>
<tr>
<td>$GODOT_CONTROLS_ENABLED</td>
<td>String true if html/controls_enabled is enabled, false otherwise</td>
</tr>
<tr>
<td>$GODOT_HEAD_TITLE</td>
<td>Title of the page, normally used as content of the HTML &lt;title&gt; element</td>
</tr>
<tr>
<td>$GODOT_HEAD_INCLUDE</td>
<td>Custom string to include just before the end of the HTML &lt;head&gt; element</td>
</tr>
<tr>
<td>$GODOT_STYLE_FONT_FAMILY</td>
<td>CSS format font-family to use, without terminating semicolon</td>
</tr>
<tr>
<td>$GODOT_STYLE_INCLUDE</td>
<td>Custom string to include just before the end of the page’s CSS</td>
</tr>
</tbody>
</table>

The first five of the placeholders listed should always be implemented in the HTML page, since they are important for the correct presentation of the game. The other placeholders are optional.

Finally, the custom HTML page is installed by replacing the existing godot.html file in the export template with the new one, retaining the name of the original.

8.10 Batch building templates

Note: This page is outdated and needs to be updated or removed.

The following is almost the same script that we use to build all the export templates that go to the website. If you want to build or roll them yourself, this might be of use.

(note: Apple stuff is missing)

```bash
# This script is intended to run on Linux or OSX. Cygwin might work.

# if this flag is set, build is tagged as release in the version
# echo $IS_RELEASE_BUILD

# Need to set path to Emscripten
export EMSCRIPTEN_ROOT=/home/to/emscripten

# Build templates
```
#remove this stuff, will be created anew
rm -rf templates
mkdir -p templates

# Windows 32 Release and Debug
scons -j 4 p=windows target=release tools=no bits=32
cp bin/godot.windows.opt.32.exe templates/windows_32_release.exe
upx templates/windows_32_release.exe
scons -j 4 p=windows target=release_debug tools=no bits=32
cp bin/godot.windows.opt.debug.32.exe templates/windows_32_debug.exe
upx templates/windows_32_debug.exe

# Windows 64 Release and Debug (UPX does not support it yet)
scons -j 4 p=windows target=release tools=no bits=64
cp bin/godot.windows.opt.64.exe templates/windows_64_release.exe
x86_64-w64-mingw32-strip templates/windows_64_release.exe
scons -j 4 p=windows target=release_debug tools=no bits=64
cp bin/godot.windows.opt.debug.64.exe templates/windows_64_debug.exe
x86_64-w64-mingw32-strip templates/windows_64_debug.exe

# Linux 64 Release and Debug
scons -j 4 p=x11 target=release tools=no bits=64
cp bin/godot.x11.opt.64 templates/linux_x11_64_release
upx templates/linux_x11_64_release
scons -j 4 p=x11 target=release_debug tools=no bits=64
cp bin/godot.x11.opt.debug.64 templates/linux_x11_64_debug
upx templates/linux_x11_64_debug

# Linux 32 Release and Debug
scons -j 4 p=x11 target=release tools=no bits=32
cp bin/godot.x11.opt.32 templates/linux_x11_32_release
upx templates/linux_x11_32_release
scons -j 4 p=x11 target=release_debug tools=no bits=32
cp bin/godot.x11.opt.debug.32 templates/linux_x11_32_debug
upx templates/linux_x11_32_debug

# Server for 32 and 64 bits (always in debug)
scons -j 4 p=server target=release_debug tools=no bits=64
cp bin/godot_server.server.opt.debug.64 templates/linux_server_64
upx templates/linux_server_64
scons -j 4 p=server target=release_debug tools=no bits=32
cp bin/godot_server.server.opt.debug.32 templates/linux_server_32
upx templates/linux_server_32

# Android
**IMPORTANT REPLACE THIS BY ACTUAL VALUES**

export ANDROID_HOME=/home/to/android-sdk
export ANDROID_NDK_ROOT=/home/to/android-ndk

# git does not allow empty dirs, so create those
mkdir -p platform/android/java/libs/armeabi

8.10. Batch building templates
mkdir -p platform/android/java/libs/x86

# Android Release
scons -j 4 p=android target=release
cp bin/libgodot.android.opt.so platform/android/java/libs/armeabi/libgodot_android.so
./gradlew build
cp platform/android/java/build/outputs/apk/java-release-unsigned.apk templates/ →android_release.apk

# Android Debug
scons -j 4 p=android target=release_debug
cp bin/libgodot.android.opt.debug.so platform/android/java/libs/armeabi/libgodot_ →android.so
./gradlew build
cp platform/android/java/build/outputs/apk/java-release-unsigned.apk templates/ →android_debug.apk

# Emscripten
scons -j 4 p=javascript target=release
cp bin/godot.javascript.opt.html godot.html
cp bin/godot.javascript.opt.js godot.js
cp tools/html_fs/filesystem.js .
zip javascript_release.zip godot.html godot.js filesystem.js
mv javascript_release.zip templates/
scons -j 4 p=javascript target=release_debug
cp bin/godot.javascript.opt.debug.html godot.html
cp bin/godot.javascript.opt.debug.js godot.js
cp tools/html_fs/filesystem.js .
zip javascript_debug.zip godot.html godot.js filesystem.js
mv javascript_debug.zip templates/

# BlackBerry 10 (currently disabled)
./path/to/bbndk/bbndk-env.sh
#scons -j 4 platform/bb10/godot_bb10_opt.qnx.armle target=release
#cp platform/bb10/godot_bb10_opt.qnx.armle platform/bb10/bar

#scons -j 4 platform/bb10/godot_bb10.qnx.armle target=release_debug
#cp platform/bb10/godot_bb10.qnx.armle platform/bb10/bar
#cd platform/bb10/bar
#zip -r bb10.zip *
#mv bb10.zip ../../../templates
#cd ../../../

# Build on Mac

[...]# Build release executables with editor
mkdir -p release
scons -j 4 p=server target=release_debug bits=64
8.11 Packaging Godot

Starting with 2.0, Godot has features to make it easier to package it for application repositories.

8.11.1 Default behaviour

Per default, Godot stores all settings and installed templates in a per-user directory. First Godot checks the APPDATA environment variable. If it exists, the per-user directory is the “Godot” subdirectory of $APPDATA. If APPDATA doesn’t exist, Godot checks the HOME environment variable. The per-user directory is then the “.godot” subdir of $HOME.

This meets common operating system standards.
8.11.2 Global template path (Unix only)

The `unix_global_settings_path` build variable is meant for Unix/Linux distro packagers who want to package export templates together with godot. It allows to put the export templates on a hardcoded path.

To use it, pass the desired path via the scons `unix_global_settings_path` build variable when building the editor. The export templates then live at the “templates” subdirectory of the path specified.

Templates installed at the per-user location still override the system wide templates.

This option is only available on unix based platforms.

8.11.3 Self contained mode

The self contained mode can be used to package godot for distribution systems where godot doesn’t live at a fixed location. If the godot editor finds a `.sc_` file in the directory the executable is located, godot will continue in “self contained mode”.

In self contained mode, all config files are located next to the executable in a directory called `editor_data`. Godot doesn’t read or write to the per-user location anymore.

The contents of the `.sc_` file (when not empty) are read with the ConfigFile api (same format as `engine.cfg`, etc). So far it can contain a list of pre-loaded project in this format:

```
[init_projects]
list=["demos/2d/platformer", "demos/2d/isometric"]
```

The paths are relative to the executable location, and will be added to the file `editor_settings.xml` when this is created for the first time.
9.1 Introduction to Godot development

This page is meant to introduce the global organization of Godot Engine’s source code, and give useful tips for extending/fixing the engine on the C++ side.

9.1.1 Architecture diagram

The following diagram describes the architecture used by Godot, from the core components down to the abstracted drivers, via the scene structure and the servers.
9.1.2 Debugging the editor with gdb

If you are writing or correcting bugs affecting Godot Engine’s editor, remember that the binary will by default run the project manager first, and then only run the editor in another process once you’ve selected a project. To launch a project directly, you need to run the editor by passing the \(-e\) argument to Godot Engine’s binary from within your project’s folder. Typically:

```
$ cd ~/myproject
$ gdb godot
```
> run -e

Or:

```bash
gdb godot
> run -e -path ~/myproject
```

## 9.2 Configuring an IDE

We assume that you already cloned and compiled Godot.

### 9.2.1 Kdevelop

It is a free, open source IDE (Integrated Development Environment) for Linux, Solaris, FreeBSD, Mac OS X and other Unix flavors.

You can find a video tutorial [here](#). Or you may follow this text version tutorial.

Start by opening Kdevelop and choosing “open project”.

Choose the directory where you cloned Godot.
For the build system, choose “custom build system”.
Now that the project has been imported, open the project configuration.
Add the following includes/imports:

```cpp
// a dot to indicate the root of the Godot project
core/
core/os/
core/math/
tools/
drivers/
platform/x11/ // make that platform/osx/ is you're using OS X
```
Apply the changes then switch to the “Custom Buildsystem” tab. Leave the build directory blank. Enable build tools and add `scons` as the executable and add `platform=x11 target=debug` (platform=osx if you're on OS X).

Next we need to tell KDevelop where to find the binary. From the “run” menu, choose “Configure Launches”.
Click “Add new” if no launcher exists. Then add the path to your executable in the executable section. Your executable should be located in the bin/ sub-directory and should be named something like godot.x11.tools.64 (the name could be different depending on your platform and depending on your build options).
That’s it! Now you should be good to go :)

### 9.2.2 Eclipse

TODO.

### 9.2.3 QtCreator

**Importing the project**

- Choose *New Project -> Import Project -> Import Existing Project*.
- Set the path to your Godot root directory and enter the project name.
- Here you can choose which folders and files will be visible to the project. C/C++ files are added automatically. Potentially useful additions: *.py for buildsystem files, *.java for Android development, *.mm for OSX. Click “Next”.
- Click *Finish*.
- Add a line containing . to *project_name.includes* to get working code completion.

**Build and run**

Build configuration:

- Click on *Projects* and open the *Build* tab.
• Delete the pre-defined `make` build step.
• Click `Add Build Step` -> `Custom Process Step`.
• Type `scons` in the `Command` field.
• Fill the `Arguments` field with your compilation options. (e.g.: `p=x11 target=debug -j 4`)

Run configuration:
• Open the `Run` tab.
• Point the `Executable` to your compiled Godot binary (e.g: `%{buildDir}\bin\godot.windows.tools.64.exe`)
• If you want to run a specific game or project, point `Working directory` to the game directory.
• If you want to run the editor, add `-e` to the `Command line arguments` field.

### 9.2.4 Xcode

#### Project Setup

• Create an Xcode external build project anywhere

<table>
<thead>
<tr>
<th>iOS</th>
<th>In-App Purchase</th>
<th>Cocoa-AppleScript</th>
<th>Empty</th>
<th>External Build System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Framework &amp; Library</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>watchOS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Application</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Framework &amp; Library</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tvOS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Application</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Framework &amp; Library</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OS X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Application</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Framework &amp; Library</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>System Plug-in</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Other</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

• Set the `Build tool` to the path to scons

Modify Build Target’s Xcode Info Tab:
• Set `Arguments` to something like: `platform=osx tools=yes bits=64 target=debug`
• Set `Directory` to the path to Godot’s source folder. Keep it blank if project is already there.
• You may uncheck `Pass build settings in environment`
Add a Command Line Target:

- Go to Xcode File > New > Target... and add a new Xcode command line target

  • Name it something so you know not to compile with this target
• e.g. GodotXcodeIndex
• Goto the newly created target’s Build Settings tab and search for Header Search Paths
• Set Header Search Paths to an absolute path to Godot’s source folder
• Make it recursive by adding two “*” s to the end of the path
• e.g. /Users/me/repos/godot-source/**

Add Godot Source to the Project:

• Drag and drop godot source into project file browser.
• Uncheck Create External Build System

• Click Next
• Select create groups
• Check off only your command line target in the Add to targets section
• Click finish. Xcode will now index the files.
• Grab a cup of coffee... Maybe make something to eat, too
• You should have jump to definition, auto completion, and full syntax highlighting when it is done.

**Scheme Setup**

Edit Build Scheme of External Build Target:

• Open scheme editor of external build target
• Expand the Build menu
• Goto Post Actions
• Add a new script run action, select your project in Provide build settings from as this allows you to use ${PROJECT_DIR} variable.

```
ln -f ${PROJECT_DIR}/godot/bin/godot.osx.tools.64 ${PROJECT_DIR}/godot/bin/godot
```

• Write a script that gives the binary a name that Xcode will recognize
• e.g. `ln -f ${PROJECT_DIR}/godot/bin/godot.osx.tools.64 ${PROJECT_DIR}/godot/bin/godot`

Build the external build target

Edit Run Scheme of External Build Target:

• Open the scheme editor again
• Click Run
• Set the Executable to the file you linked in your post build action script
• Check Debug executable if it isn’t already
• You can go to Arguments tab and add an -e and a -path to a project to debug the editor not the project selection screen

Test it:
• Set a breakpoint in platform/osx/godot_main_osx.mm
• It should break at the point!

9.2.5 Other editors (vim, emacs, Atom...)

TODO.

9.3 Core types

Godot has a rich set of classes and templates that compose its core, and everything is built upon them.

This reference will try to list them in order for their better understanding.

9.3.1 Definitions

Godot uses the standard C98 datatypes, such as uint8_t, uint32_t, int64_t, etc. which are nowadays supported by every compiler. Reinventing the wheel for those is not fun, as it makes code more difficult to read.

In general, care is not taken to use the most efficient datatype for a given task unless using large structures or arrays. int is used through most of the code unless necessary. This is done because nowadays every device has at least a 32 bits bus and can do such operations in one cycle. It makes code more readable too.

For files or memory sizes, size_t is used, which is warranted to be 64 bits.

For Unicode characters, CharType instead of wchar_t is used, because many architectures have 4 bytes long wchar_t, where 2 bytes might be desired. However, by default, this has not been forced and CharType maps directly to wchar_t.
References:

- core/typedefs.h

9.3.2 Memory model

PC is a wonderful architecture. Computers often have gigabytes of RAM, terabytes of storage and gigahertz of CPU, and when an application needs more resources the OS will just swap out the inactive ones. Other architectures (like mobile or consoles) are in general more limited.

The most common memory model is the heap, where an application will request a region of memory, and the underlying OS will try to fit it somewhere and return it. This often works best and is very flexible, but over time and with abuse, this can lead to segmentation.

Segmentation slowly creates holes that are too small for most common allocations, so that memory is wasted. There is a lot of literature about heap and segmentation, so this topic will not be developed further here. Modern operating systems use paged memory, which helps mitigate the problem of segmentation but doesn’t solve it.

However, in many studies and tests, it is shown that given enough memory, if the maximum allocation size is below a given threshold in proportion to the maximum heap size and proportion of memory intended to be unused, segmentation will not be a problem over time as it will remain constant. In other words, just leave 10-20% of your memory free and perform all small allocations and you are fine.

Godot ensures that all objects that can be allocated dynamically are small (less than a few kb at most). But what happens if an allocation is too large (like an image or mesh geometry or large array)? In this case Godot has the option to use a dynamic memory pool. This memory needs to be locked to be accessed, and if an allocation runs out of memory, the pool will be rearranged and compacted on demand. Depending on the need of the game, the programmer can configure the dynamic memory pool size.

9.3.3 Allocating memory

Godot has many tools for tracking memory usage in a game, specially during debug. Because of this, the regular C and C++ library calls should not be used. Instead, a few other ones are provided.

For C-style allocation, Godot provides a few macros:

```cpp
memalloc()
memrealloc()
memfree()
```

These are equivalent to the usual malloc, realloc, free of the standard library.

For C++-style allocation, special macros are provided:

```cpp
memnew( Class / Class(args) )
memdelete( instance )
memnew_arr( Class , amount )
memdelete_arr( pointer to array )
```

which are equivalent to new, delete, new[] and delete[].

memnew/memdelete also use a little C++ magic and notify Objects right after they are created, and right before they are deleted.

For dynamic memory, the DVector<> template is provided. Just use it like:
DVector<int>

DVector is just a standard vector class, it can be accessed using the [] operator, but that’s probably slow for large amount of accesses (as it has to lock internally). A few helpers exist for this:

```cpp
DVector<int>::Read r = dvector.read()
int someint = r[4]
```

and

```cpp
DVector<int>::Write w = dvector.write()
w[4]=22;
```

respectively. These allow fast read/write from DVectors and keep it locked until they go out of scope.

References:

- core/os/memory.h
- core/dvector.h

9.3.4 Containers

Godot provides also a set of common containers:

- Vector
- List
- Set
- Map

They are very simple and aim to be as minimal as possible, as templates in C++ are often inlined and make the binary size much fatter, both in debug symbols and code. List, Set and Map can be iterated using pointers, like this:

```cpp
for(List<int>::Element *E=somelist.front();E;E=E->next()) {
    print_line(E->get()); //print the element
}
```

The Vector<> class also has a few nice features:

- It does copy on write, so making copies of it is cheap as long as they are not modified.
- It supports multi-threading, by using atomic operations on the reference counter.

References:

- core/vector.h
- core/list.h
- core/set.h
- core/map.h
9.3.5 String

Godot also provides a String class. This class has a huge amount of features, full Unicode support in all the functions (like case operations) and utf8 parsing/extracting, as well as helpers for conversion and visualization.

References:
- core/ustring.h

9.3.6 StringName

StringNames are like a String, but they are unique. Creating a StringName from a string results in a unique internal pointer for all equal strings. StringNames are really useful for using strings as identifier, as comparing them is basically comparing a pointer.

Creation of a StringName (specially a new one) is slow, but comparison is fast.

References:
- core/string_db.h

9.3.7 Math types

There are several linear math types available in the core/math directory, they are basically just that.

References:
- core/math

9.3.8 NodePath

This is a special datatype used for storing paths in a scene tree and referencing them fast.

References:
- core/path_db.h

9.3.9 RID

RIDs are resource IDs. Servers use these to reference data stored in them. RIDs are opaque, meaning that the data they reference can't be accessed directly. RIDs are unique, even for different types of referenced data.

References:
- core/rid.h
9.4 Variant class

9.4.1 About

Variant is the most important datatype of Godot, it’s the most important class in the engine. A Variant takes up only 20 bytes and can store almost any engine datatype inside of it. Variants are rarely used to hold information for long periods of time, instead they are used mainly for communication, editing, serialization and generally moving data around.

A Variant can:

- Store almost any datatype
- Perform operations between many variants (GDScript uses Variant as its atomic/native datatype).
- Be hashed, so it can be compared quickly to over variants
- Be used to convert safely between datatypes
- Be used to abstract calling methods and their arguments (Godot exports all its functions through variants)
- Be used to defer calls or move data between threads.
- Be serialized as binary and stored to disk, or transferred via network.
- Be serialized to text and use it for printing values and editable settings.
- Work as an exported property, so the editor can edit it universally.
- Be used for dictionaries, arrays, parsers, etc.

Basically, thanks to the Variant class, writing Godot itself was a much, much easier task, as it allows for highly dynamic constructs not common of C++ with little effort. Become a friend of Variant today.

References:

- core/variant.h

9.4.2 Dictionary and Array

Both are implemented using variants. A Dictionary can match any datatype used as key to any other datatype. An Array just holds an array of Variants. Of course, a Variant can also hold a Dictionary and an Array inside, making it even more flexible.

Both have a shared mode and a COW mode. Scripts often use them in shared mode (meaning modifications to a container will modify all references to it), or COW mode (modifications will always alter the local copy, making a copy of the internal data if necessary, but will not affect the other copies). In COW mode, Both Dictionary and Array are thread-safe, otherwise a Mutex should be created to lock if multi thread access is desired.

References:

- core/dictionary.h
- core/array.h
9.5 Object class

9.5.1 General definition

`Object` is the base class for almost everything. Most classes in Godot inherit directly or indirectly from it. Objects provide reflection and editable properties, and declaring them is a matter of using a single macro like this.

```cpp
class CustomObject : public Object {
    OBJ_TYPE(CustomObject, Object); // this is required to inherit
};
```

This makes Objects gain a lot of functionality, like for example

```cpp
obj = memnew(CustomObject);
print_line("Object Type: ", obj->get_type()); //print object type
obj2 = obj->cast_to<OtherType>(); // converting between types, this also works without RTTI enabled.
```

References:

- core/object.h

9.5.2 Registering an Object

`ObjectTypeDB` is a static class that holds the entire list of registered classes that inherit from `Object`, as well as dynamic bindings to all their methods properties and integer constants.

Classes are registered by calling:

```cpp
ObjectTypeDB::register_type<MyCustomType>();
```

Registering it will allow the type to be instanced by scripts, code, or creating them again when deserializing. Registering as virtual is the same but it can’t be instanced.

```cpp
ObjectTypeDB::register_virtual_type<MyCustomType>();
```

Object-derived classes can override the static function `static void _bind_methods()`. When one class is registered, this static function is called to register all the object methods, properties, constants, etc. It’s only called once. If an Object derived class is instanced but has not been registered, it will be registered as virtual automatically.

Inside `_bind_methods`, there are a couple of things that can be done. Registering functions is one:

```cpp
ObjectTypeDB::register_method(_MD("methodname","arg1name","arg2name"),&MyCustomMethod);
```

Default values for arguments can be passed in reverse order:

```cpp
ObjectTypeDB::register_method(_MD("methodname","arg1name","arg2name"),&MyCustomType::_method,DEFVAL(-1)); //default value for arg2name
```
_MD is a macro that converts “methodname” to a StringName for more efficiency. Argument names are used for introspection, but when compiling on release, the macro ignores them, so the strings are unused and optimized away. Check _bind_methods of Control or Object for more examples.

If just adding modules and functionality that is not expected to be documented as thoroughly, the _MD() macro can safely be ignored and a string passing the name can be passed for brevity.

References:

- core/object_type_db.h

### 9.5.3 Constants

Classes often have enums such as:

```cpp
enum SomeMode {
    MODE_FIRST,
    MODE_SECOND
};
```

For these to work when binding to methods, the enum must be declared convertible to int, for this a macro is provided:

```cpp
VARIANT_ENUM_CAST( MyClass::SomeMode ); // now functions that take SomeMode can be bound.
```

The constants can also be bound inside _bind_methods, by using:

```cpp
BIND_CONSTANT( MODE_FIRST );
BIND_CONSTANT( MODE_SECOND );
```

### 9.5.4 Properties (set/get)

Objects export properties, properties are useful for the following:

- Serializing and deserializing the object.
- Creating a list of editable values for the Object derived class.

Properties are usually defined by the PropertyInfo() class. Usually constructed as:

```cpp
PropertyInfo(type,name,hint,hint_string,usage_flags)
```

For example:

```cpp
PropertyInfo(Variant::INT,"amount",PROPERTY_HINT_RANGE,"0,49,1",PROPERTY_USAGE_EDITOR)
```

This is an integer property, named “amount”, hint is a range, range goes from 0 to 49 in steps of 1 (integers). It is only usable for the editor (edit value visually) but won’t be serialized.

Another example:

```cpp
PropertyInfo(Variant::STRING,"modes",PROPERTY_HINT_ENUM,"Enabled,Disabled,Turbo")
```
This is a string property, can take any string but the editor will only allow the defined hint ones. Since no usage flags were specified, the default ones are PROPERTY_USAGE_STORAGE and PROPERTY_USAGE_EDITOR.

There are plenty of hints and usage flags available in object.h, give them a check.

Properties can also work like C# properties and be accessed from script using indexing, but this usage is generally discouraged, as using functions is preferred for legibility. Many properties are also bound with categories, such as “animation/frame” which also make indexing impossible unless using operator [].

From _bind_methods(), properties can be created and bound as long as set/get functions exist. Example:

```
ADD_PROPERTY( PropertyInfo(Variant::INT,"amount"), _SCS("set_amount"), _SCS("get_amount") )
```

This creates the property using the setter and the getter. _SCS is a macro that creates a StringName efficiently.

### 9.5.5 Binding properties using _set/_get/_get_property_list

An additional method of creating properties exists when more flexibility is desired (i.e. adding or removing properties on context).

The following functions can be overridden in an Object derived class, they are NOT virtual, DO NOT make them virtual, they are called for every override and the previous ones are not invalidated (multilevel call).

```
void _get_property_info(List<PropertyInfo> *r_props); // return list of properties
bool _get(const StringName& p_property, Variany& r_value) const; // return true if _property was found
bool _set(const StringName& p_property, const Variany& p_value); // return true if _property was found
```

This is also a little less efficient since p_property must be compared against the desired names in serial order.

### 9.5.6 Dynamic casting

Godot provides dynamic casting between Object-derived classes, for example:

```
void somefunc(Object *some_obj) {
    Button *button = some_obj->cast_to<Button>();
}
```

If cast fails, NULL is returned. This system uses RTTI, but it also works fine (although a bit slower) when RTTI is disabled. This is useful on platforms where a very small binary size is ideal, such as HTML5 or consoles (with low memory footprint).

### 9.5.7 Signals

Objects can have a set of signals defined (similar to Delegates in other languages). Connecting to them is rather easy:

```
obj->connect("signal",target_instance,target_method)
// for example
obj->connect("enter_tree",this,"_node_entered_tree")
```

The method _node_entered_tree must be registered to the class using ObjectTypeDB::register_method (explained before).
Adding signals to a class is done in `bind_methods`, using the `ADD_SIGNAL` macro, for example:

```cpp
ADD_SIGNAL( MethodInfo("been_killed") )
```

### 9.5.8 References

*Reference* inherits from Object and holds a reference count. It is the base for reference counted object types. Declaring them must be done using `Ref<>` template. For example:

```cpp
class MyReference : public Reference {
    OBJ_TYPE( MyReference,Reference );
};
Ref<MyReference> myref = memnew( MyReference );
```

`myref` is reference counted. It will be freed when no more `Ref<>` templates point to it.

**References:**

- core/reference.h

### 9.5.9 Resources:

*Resource* inherits from Reference, so all resources are reference counted. Resources can optionally contain a path, which reference a file on disk. This can be set with `resource.set_path(path)`. This is normally done by the resource loader though. No two different resources can have the same path, attempt to do so will result in an error.

Resources without a path are fine too.

**References:**

- core/resource.h

### 9.5.10 Resource loading

Resources can be loaded with the ResourceLoader API, like this:

```cpp
Ref<Resource> res = ResourceLoader::load("res://someresource.res")
```

If a reference to that resource has been loaded previously and is in memory, the resource loader will return that reference. This means that there can be only one resource loaded from a file referenced on disk at the same time.

- resourceinteractiveloader (TODO)

**References:**

- core/io/resource_loader.h
9.5.11 Resource saving

Saving a resource can be done with the resource saver API:

```cpp
ResourceSaver::save("res://someresource.res", instance)
```

Instance will be saved. Sub resources that have a path to a file will be saved as a reference to that resource. Sub resources without a path will be bundled with the saved resource and assigned sub-IDs, like “res://someresource.res::1”. This also helps to cache them when loaded.

References:

- core/io/resource_saver.h

9.6 Inheritance class tree

9.6.1 Object
9.6.3 Control

9.6.4 Node2D
9.7 Custom modules in C++

9.7.1 Modules

Godot allows extending the engine in a modular way. New modules can be created and then enabled/disabled. This allows for adding new engine functionality at every level without modifying the core, which can be split for use and reuse in different modules.

Modules are located in the modules/ subdirectory of the build system. By default, two modules exist, GDScript (which, yes, is not part of the core engine), and the GridMap. As many new modules as desired can be created and combined, and the SCons build system will take care of it transparently.
9.7.2 What for?

While it’s recommended that most of a game is written in scripting (as it is an enormous time saver), it’s perfectly possible to use C++ instead. Adding C++ modules can be useful in the following scenarios:

- Binding an external library to Godot (like Bullet, Physx, FMOD, etc).
- Optimize critical parts of a game.
- Adding new functionality to the engine and/or editor.
- Porting an existing game.
- Write a whole, new game in C++ because you can’t live without C++.

9.7.3 Creating a new module

Before creating a module, make sure to download the source code of Godot and manage to compile it. There are tutorials in the documentation for this.

To create a new module, the first step is creating a directory inside `modules/`. If you want to maintain the module separately, you can checkout a different VCS into modules and use it.

The example module will be called “sumator”, and is placed inside the Godot source tree (C:\godot refers to wherever the Godot sources are located):

```bash
C:\godot> cd modules
C:\godot\modules> mkdir sumator
C:\godot\modules> cd sumator
C:\godot\modules\sumator>
```

Inside we will create a simple sumator class:

```c
/* sumator.h */
#ifndef SUMATOR_H
#define SUMATOR_H

#include "reference.h"

class Sumator : public Reference {
    OBJ_TYPE(Sumator,Reference);

    int count;

protected:
    static void _bind_methods();

public:
    void add(int value);
    void reset();
    int get_total() const;

    Sumator();
};
#endif
```

And then the cpp file.
Then, the new class needs to be registered somehow, so two more files need to be created:

```cpp
/* register_types.h */
void register_sumator_types();
void unregister_sumator_types();
/* yes, the word in the middle must be the same as the module folder name */

/* register_types.cpp */
#include "register_types.h"
#include "object_type_db.h"
#include "sumator.h"

void register_sumator_types() {
    ObjectTypeDB::register_type<Sumator>();
}
void unregister_sumator_types() {
    //nothing to do here
}
```

9.7. Custom modules in C++
Next, we need to create a `SCsub` file so the build system compiles this module:

```python
# SCsub
Import('env')

env.add_source_files(env.modules_sources, '*.cpp')  # just add all cpp files to the build
```

If you want to add custom compiler flags when building your module, you need to clone `env` first, so it won’t add those flags to whole Godot build (which can cause errors). Example `SCsub` with custom flags:

```python
# SCsub
Import('env')

module_env = env.Clone()
module_env.add_source_files(env.modules_sources, '*.cpp')
module_env.Append(CXXFLAGS=['-O2', '-std=c++11'])
```

And finally, the configuration file for the module, this is a simple python script that must be named `config.py`:

```python
# config.py

def can_build(platform):
    return True

def configure(env):
    pass
```

The module is asked if it’s ok to build for the specific platform (in this case, True means it will build for every platform).

And that’s it. Hope it was not too complex! Your module should look like this:

```
godot/modules/sumator/config.py
godot/modules/sumator/sumator.h
godot/modules/sumator/sumator.cpp
godot/modules/sumator/register_types.h
godot/modules/sumator/register_types.cpp
godot/modules/sumator/SCsub
```

You can then zip it and share the module with everyone else. When building for every platform (instructions in the previous sections), your module will be included.

### 9.7.4 Using the module

Using your newly created module is very easy, from any script you can now do:

```python
var s = Sumator.new()
s.add(10)
s.add(20)
s.add(30)
print(s.get_total())
s.reset()
```

And the output will be 60.
9.7.5 Improving the build system for development

So far we defined a clean and simple SCsub that allows us to add the sources of our new module as part of the Godot binary.

This static approach is fine when we want to build a release version of our game given we want all the modules in a single binary.

However the trade-off is every single change means a full recompilation of the game. Even if SCons is able to detect and recompile only the file that have changed, finding such files and eventually linking the final binary is a long and costly part.

The solution to avoid such a cost is to build our own module as a shared library that will be dynamically loaded when starting our game’s binary.

```python
# SCsub
Import('env')

sources = [
    "register_types.cpp",
    "sumator.cpp"
]

# First, create a custom env for the shared library.
module_env = env.Clone()
module_env.Append(CXXFLAGS='-fPIC')  # Needed to compile shared library
# We don't want godot's dependencies to be injected into our shared library.
module_env['LIBS'] = []

# Now define the shared library. Note that by default it would be built
# into the module’s folder, however it’s better to output it into `bin`
# next to the godot binary.
shared_lib = module_env.SharedLibrary(target='#bin/sumator', source=sources)

# Finally notify the main env it has our shared library as a new dependency.
# To do so, SCons wants the name of the lib with it custom suffixes
# (e.g. "x11tools.64") but without the final ".so".
# We pass this along with the directory of our library to the main env.
shared_lib_shim = shared_lib[0].name.rsplit('.', 1)[0]
env.Append(LIBS=[shared_lib_shim])
env.Append(LIBPATH=['#bin'])
```

Once compiled, we should end up with a bin directory containing both the godot* binary and our libsumator*.so. However given the .so is not in a standard directory (like /usr/lib), we have to help our binary find it during runtime with the LD_LIBRARY_PATH environ variable:

```
user@host:~/godot$ export LD_LIBRARY_PATH=`pwd`/bin/
user@host:~/godot$ ./bin/godot*
```

Note: Pay attention you have to export the environ variable otherwise you won’t be able to play you project from within the editor.

On top of that, it would be nice to be able to select whether to compile our module as shared library (for development) or as a part of the godot binary (for release). To do that we can define a custom flag to be passed to SCons using the ARGUMENT command:

```python
# SCsub
Import('env')
```
sources = [
    "register_types.cpp",
    "sumator.cpp"
]

module_env = env.Clone()
module_env.Append(CXXFLAGS=['-O2', '-std=c++11'])

if ARGUMENTS.get('sumator_shared', 'no') == 'yes':
    # Shared lib compilation
    module_env.Append(CXXFLAGS='-fPIC')
    module_env['LIBS'] = []
    shared_lib = module_env.SharedLibrary(target='#bin/sumator', source=sources)
    shared_lib_shim = shared_lib[0].name.rsplit('.', 1)[0]
    env.Append(LIBS=[shared_lib_shim])
    env.Append(LIBPATH=['#bin'])
else:
    # Static compilation
    module_env.add_source_files(env.modules_sources, sources)

Now by default scons command will build our module as part of godot’s binary and as a shared library when passing sumator_shared=yes.

Finally you can even speedup build further by explicitly specifying your shared module as target in the scons command:

```
user@host:~/godot $ scons sumator_shared=yes bin/sumator.x11.tools.64.so
```

### 9.7.6 Summing up

As you see, it’s really easy to develop Godot in C++. Just write your stuff normally and remember to:

- use **OBJ_TYPE** macro for inheritance, so Godot can wrap it
- use **_bind_methods** to bind your functions to scripting, and to allow them to work as callbacks for signals.

But this is not all, depending what you do, you will be greeted with some surprises.

- If you inherit from **Node** (or any derived node type, such as Sprite), your new class will appear in the editor, in the inheritance tree in the “Add Node” dialog.
- If you inherit from **Resource**, it will appear in the resource list, and all the exposed properties can be serialized when saved/loaded.
- By this same logic, you can extend the Editor and almost any area of the engine.

### 9.8 Creating Android modules

#### 9.8.1 Introduction

Making video games portable is all fine and dandy, until mobile gaming monetization shows up.

This area is complex, usually a mobile game that monetizes needs special connections to a server for stuff such as:

- Analytics
- In-app purchases
• Receipt validation
• Install tracking
• Ads
• Video ads
• Cross-promotion
• In-game soft & hard currencies
• Promo codes
• A/B testing
• Login
• Cloud saves
• Leaderboards and scores
• User support & feedback
• Posting to Facebook, Twitter, etc.
• Push notifications

Oh yeah, developing for mobile is a lot of work. On iOS, you can just write a C++ module and take advantage of the C++/ObjC intercommunication, so this is rather easy.

For C++ developers Java is a pain, the build system is severely bloated and interfacing it with C++ through JNI (Java Native Interface) is more pain that you don’t want even for your worst enemy.

9.8.2 Maybe REST?

Most of these APIs allow communication via REST+JSON APIs. Godot has great support for HTTP, HTTPS and JSON, so consider this as an option that works in every platform. Only write the code once and you are set to go.

Popular engines that have half the share of apps published on mobile get special plugins written just for them. Godot does not have that luxury yet. So, if you write a REST implementation of a SDK for Godot, please share it with the community.

9.8.3 Android module

Writing an Android module is similar to Custom modules in C++, but needs a few more steps.

Make sure you are familiar with building your own Android export templates, as well as creating Custom modules in C++.

config.py

In the config.py for the module, some extra functions are provided for convenience. First, it’s often wise to detect if android is being built and only enable building in this case:

```python
def can_build(plat):
    return plat=="android"
```

If more than one platform can be built (typical if implementing the module also for iOS), check manually for Android in the configure functions:
9.8.4 Java singleton

An android module will usually have a singleton class that will load it, this class inherits from Godot.SingletonBase. Resource identifiers for any additional resources you have provided for the module will be in the `com.godot.game.R` class, so you’ll likely want to import it.

A singleton object template follows:

```java
// package com.android.godot; // for 1.1
package org.godotengine.godot; // for 2.0
import com.godot.game.R;

public class MySingleton extends Godot.SingletonBase {
    public int myFunction(String p_str) {
        // a function to bind
    }

    static public Godot.SingletonBase initialize(Activity p_activity) {
        return new MySingleton(p_activity);
    }

    public MySingleton(Activity p_activity) {
        // register class name and functions to bind
        registerClass("MySingleton", new String[]{"myFunction"});

        // you might want to try initializing your singleton here, but android
        // threads are weird and this runs in another thread, so you usually have to...
        activity.runOnUiThread(new Runnable() {
            public void run() {
                // useful way to get config info from engine.cfg
                String key = GodotLib.getGlobal("plugin/api_key");
                SDK.initializeHere();

            }
        });
    }

    // forwarded callbacks you can reimplement, as SDKs often need them
    protected void onMainActivityResult(int requestCode, int resultCode, Intent data) {}
    protected void onMainPause() {}
    protected void onMainResume() {}
    protected void onMainDestroy() {}
```
Calling back to Godot from Java is a little more difficult. The instance ID of the script must be known first, this is obtained by calling `get_instance_ID()` on the script. This returns an integer that can be passed to Java.

From Java, use the `calldeferred` function to communicate back with Godot. Java will most likely run in a separate thread, so calls are deferred:

```java
GodotLib.calldeferred(<instanceid>, "<function>", new Object[]{param1,param2,etc});
```

Add this singleton to the build of the project by adding the following to config.py:

(Before Version 2.0)

```python
def can_build(plat):
    return plat=="android" or plat=="iphone"

def configure(env):
    if env['platform'] == 'android':
        # will copy this to the java folder
        env.android_module_file("MySingleton.java")
        #env.android_module_file("MySingleton2.java") call again for more files
```

(After Version 2.0)

```python
def can_build(plat):
    return plat=="android" or plat=="iphone"

def configure(env):
    if env['platform'] == 'android':
        # will copy this to the java folder
        env.android_add_java_dir("Directory that contain MySingleton.java")
```

### 9.8.5 AndroidManifest

Some SDKs need custom values in AndroidManifest.xml. Permissions can be edited from the godot exporter so there is no need to add those, but maybe other functionalities are needed.

Create the custom chunk of android manifest and put it inside the module, add it like this:

(Before Version 2.0)

```python
def can_build(plat):
    return plat=="android" or plat=="iphone"

def configure(env):
    if env['platform'] == 'android':
        # will copy this to the java folder
        env.android_module_file("MySingleton.java")
        env.android_module_manifest("AndroidManifestChunk.xml")
```

(After Version 2.0)

```python
def can_build(plat):
    return plat=="android" or plat=="iphone"

def configure(env):
    if env['platform'] == 'android':
        # will copy this to the java folder
        env.android_add_java_dir("Directory that contain MySingleton.java")
        env.android_module_manifest("AndroidManifestChunk.xml")
```
def can_build(plat):
    return plat=="android" or plat=="iphone"

def configure(env):
    if env["platform"] == 'android':
        # will copy this to the java folder
        env.android_add_java_dir("Directory that contain MySingelton.java")
        env.android_add_to_manifest("AndroidManifestChunk.xml")

9.8.6 Resources

In order to provide additional resources with your module you have to add something like this:

def configure(env):
    if env["platform"] == 'android':
        # [...]
        env.android_add_res_dir("Directory that contains resource subdirectories, \n\t\t\tvalues, drawable, etc.")

Now you can refer to those resources by their id (R.string.my_string, and the like) by importing the com.godot.game.R class in your Java code.

9.8.7 SDK library

So, finally it’s time to add the SDK library. The library can come in two flavors, a JAR file or an Android project for ant. JAR is the easiest to integrate, just put it in the module directory and add it:

(After Version 2.0)

def can_build(plat):
    return plat=="android" or plat=="iphone"

def configure(env):
    if env["platform"] == 'android':
        # will copy this to the java folder
        env.android_add_java_dir("Directory that contain MySingelton.java")
        env.android_add_to_manifest("AndroidManifestChunk.xml")
        env.android_add_dependency("compile files('something_local.jar')") # if you have a jar, the path is relative to platform/android/java/gradlew, so it will start with ../../../modules/module_name/
        env.android_add_maven_repository("maven url") #add a maven url
        env.android_add_dependency("compile 'com.google.android.gms:play-services-ads:\n\t\t\t8'") #get dependency from maven repository

(Before Version 2.0)

def can_build(plat):
    return plat=="android" or plat=="iphone"

def configure(env):
    if env["platform"] == 'android':
        # will copy this to the java folder
        env.android_module_file("MySingleton.java")
        env.android_module_manifest("AndroidManifestChunk.xml")
        env.android_module_library("MyLibrary-3.1.jar")

458 Chapter 9. Developing in C++
9.8.8 SDK project

When this is an Android project, things usually get more complex. Copy the project folder inside the module directory and configure it:

```
c:\godot\modules\mymodule\sdk-1.2> android -p . -t 15
```

As of this writing, Godot uses minSdk 10 and targetSdk 15. If this ever changes, it should be reflected in the manifest template: `AndroidManifest.xml.template`

Then, add the module folder to the project:

(Before Version 2.0)

```python
def can_build(plat):
    return plat="android" or plat="iphone"

def configure(env):
    if env['platform'] == 'android':
        # will copy this to the java folder
        env.android_module_file("MySingleton.java")
        env.android_module_manifest("AndroidManifestChunk.xml")
        env.android_module_source("sdk-1.2","")
```

(After Version 2.0)

9.8.9 Building

As you probably modify the contents of the module, and modify your .java inside the module, you need the module to be built with the rest of Godot, so compile android normally.

```
c:\godot> scons p=android
```

This will cause your module to be included, the .jar will be copied to the java folder, the .java will be copied to the sources folder, etc. Each time you modify the .java, scons must be called.

Afterwards, just continue the steps for compiling android *Compiling for Android*.

Using the module

To use the module from GDScript, first enable the singleton by adding the following line to engine.cfg (Godot Engine 2.0 and greater):

```ini
[android]
modules="org/godotengine/godot/MySingleton"
```

For Godot Engine 1.1 is

```ini
[android]
modules="com/android/godot/MySingleton"
```

More than one singleton module can be enabled by separating with commas:
Then just request the singleton Java object from Globals like this:

```
# in any file
var singleton = null
func _init():
    singleton = Globals.get_singleton("MySingleton")
    print(singleton.myFunction("Hello"))
```

### 9.8.10 Troubleshooting

(This section is a work in progress, report your problems here!)

**Godot crashes upon load**

Check `adb logcat` for possible problems, then:

- Make sure `libgodot_android.so` is in the `libs/armeabi` folder
- Check that the methods used in the Java singleton only use simple Java datatypes, more complex ones are not supported.

### 9.8.11 Future

Godot has an experimental Java API Wrapper that allows to use the entire Java API from GDScript.

It’s simple to use and it’s used like this:

```
class JavaClassWrapper.wrap(<javaclass as text>)
```

This is most likely not functional yet, if you want to test it and help us make it work, contact us through the developer mailing list.
10.1 Ways to contribute

Godot Engine is a non-profit, community-driven free and open source project. Almost all (but our lead dev Juan, more on that below) developers are working pro bono on their free time, out of personal interest and for the love of creating a libre engine of exceptional quality.

This means that to thrive, Godot needs as many users as possible to get involved by contributing to the engine. There are many ways to contribute to such a big project, making it possible for everybody to bring something positive to the engine, regardless of their skillset:

• **Be part of the community.** The best way to contribute to Godot and help it become ever better is simply to use the engine and make its promotion by word-of-mouth, in-game credits and splash screen, blog posts, tutorials, videos, demos, gamedev or free software events, support on the Q&A, IRC, forums, Discord, etc. Being a user and advocate helps spread the word about our great engine, which has no marketing budget and can therefore only rely on its community to become more mainstream.

• **Make games.** That’s no secret, to convince new users and especially industry players that Godot is a relevant market player, we need great games made with Godot. We know that the engine has a lot of potential, both for 2D and 3D games, but given its young age we still lack big releases that will draw attention to Godot. So keep working on your awesome projects, each new game increases our credibility on the gamedev market!

• **Get involved in the engine’s development.** This can be by contributing code via pull requests, testing the development snapshots or directly the git master branch, report bugs or suggest enhancements on the issue tracker, improve the official documentation (both the class reference and tutorials). The following sections will cover each of those “direct” ways of contributing to the engine.

• **Donate.** Godot is a non-profit project, but it can still benefit from user donations for many things. Apart from usual expenses such as hosting costs or promotion material on events, we also use donation money to acquire hardware when necessary (e.g. we used donation money to buy a Macbook Pro to implement Retina/HiDPI support and various other macOS-related features). Most importantly, we also used donation money to hire our lead developer Juan Linietsky, so that he can work full-time on the engine. Even with a low monthly wage, we need a steady donation income to continue doing this, which has been very beneficial to the project so far. So if you want to donate some money to the project, check our website for details.

10.1.1 Contributing code

The possibility to study, use, modify and redistribute modifications of the engine’s source code are the fundamental rights that Godot’s license grant you, making it free and open source software.

As such, everyone is entitled to modify Godot’s source code, and send those modifications back to the upstream project in the form of a patch (a text file describing the changes in a ready-to-apply manner) or - in the modern workflow that
we use - via a so-called “pull request” (PR), i.e. a proposal to directly merge one or more git commits (patches) into the main development branch.

Contributing code changes upstream has two big advantages:

• Your own code will be reviewed and improved by other developers, and will be further maintained directly in the upstream project, so you won’t have to reapply your own changes every time you move to a newer version. On the other hand it comes with a responsibility, as your changes have to be generic enough to be beneficial to all users, and not just your project; so in some cases it might still be relevant to keep your changes only for your own project, if they are too specific.

• The whole community will benefit from your work, and other contributors will behave the same way, contributing code that will be beneficial to you. At the time of this writing, more than 300 developers have contributed code changes to the engine!

To ensure a good collaboration and overall quality, the Godot developers enforce some rules for code contribution, for example regarding the style to use in the C++ code (indentation, brackets, etc.) or the git and PR workflow.

See also:

Technical details about the PR workflow are outlined in a specific section, Pull request workflow.

Details about the code style guidelines and the clang-format tool used to enforce them are outlined in Code style guidelines.

10.1.2 Testing and reporting issues

Another great way of contributing to the engine is to test development releases or the development branch and to report issues. It is also meaningful to report issues discovered in so-called stable releases, so that they can be fixed in the development branch and in future maintenance releases.

Testing development versions

To help with the testing, you have several possibilities:

• Compile the engine from source yourself, following the instructions of the Compiling page for your platform.

• Test official pre-release binaries when they are announced (usually on the blog and other community platforms), such as alpha, beta and RC (release candidate) builds.

• Test “trusted” unofficial builds of the development branch; just ask community members for reliable providers. Whenever possible, it’s best to use official binaries or to compile yourself though, to be sure about the provenance of your binaries.

As mentioned previously, it is also meaningful to keep your eyes peeled for potential bugs that might still be present in the stable releases, especially when using some niche features of the engine which might get less testing by the developers.

Filing an issue on GitHub

Godot uses GitHub’s issue tracker for bug reports and enhancement suggestions. You will need a GitHub account to be able to open a new issue there, and click on the “New issue” button.

When you report a bug, you should keep in mind that the process is very similar to an appointment with your doctor. You noticed symptoms that make you think that something might be wrong (the engine crashes, some features don’t work as expected, etc.). It’s the role of the bug triaging team and the developers to then help make the diagnosis of the issue you met, so that the actual cause of the bug can be identified and addressed.
You should therefore always ask yourself: what are the relevant information to give so that other Godot contributors can understand the bug, identify it and hopefully fix it. Here are some of the most important infos that you should always provide:

- **Operating system.** Sometimes bugs are system-specific, i.e. they happen only on Windows, or only on Linux, etc. That’s particularly relevant for all bugs related to OS interfaces, such as file management, input, window management, audio, etc.

- **Godot version.** This is a must have. Some issues might be relevant in the current stable release, but fixed in the development branch, or the other way around. You might also be using an obsolete version of Godot and experiencing a known issue fixed in a later version, so knowing this from the start helps to speed up the diagnosis.

- **How to reproduce the bug.** In the majority of cases, bugs are reproducible, i.e. it is possible to trigger them reliably by following some steps. Please always describe those steps as clearly as possible, so that everyone can try to reproduce the issue and confirm it. Ideally, make a demo project that reproduces this issue out of the box, zip it and attach it to the issue (you can do this by drag and drop). Even if you think that the issue is trivial to reproduce, adding a minimal project that lets reproduce it is a big added value. You have to keep in mind that there are thousands of issues in the tracker, and developers can only dedicate little time to each issue.

When you click the “New issue” button, you should be presented with a text area prefilled with our issue template. Please try to follow it so that all issues are consistent and provide the required information.

### 10.1.3 Contributing to the documentation

There are two separate resources referred to as “documentation” in Godot:

- **The class reference.** This is the documentation for the complete Godot API as exposed to GDScript and the other scripting languages. It can be consulted offline, directly in Godot’s code editor, or online at [Godot API](https://godot-engine.org/api). To contribute to the class reference, you have to edit the `doc/base/classes.xml` in Godot’s git repository, and make a pull request. See [Updating the class reference](https://github.com/godotengine/godot/blob/master/doc/base/classes.xml) for more details.

- **The tutorials and engine documentation.** This is the part you are reading now, which is distributed in the HTML, PDF and EPUB formats. Its contents are generated from plain text files in the reStructured Text (rst) format, to which you can contribute via pull requests on the godot-docs GitHub repository. See [Documentation guidelines](https://github.com/godotengine/godot/blob/master/doc/development/documentation-guidelines.rst) for more details.

### 10.2 Pull request workflow

The so-called “PR workflow” used by Godot is common to many projects using Git, and should be familiar to veteran free software contributors. The idea is that only a small number (if any) commit directly to the `master` branch. Instead, contributors fork the project (i.e. create a copy of it, which they can modify as they wish), and then use the GitHub interface to request a pull from one of their fork’s branch to one branch of the original (often named `upstream`) repository.

The resulting pull request (PR) can then be reviewed by other contributors, which might approve it, reject it, or most often request that modifications be done. Once approved, the PR can then be merged by one of the core developers, and its commit(s) will become part of the target branch (usually the `master` branch).

We will go through an example to show the typical workflow and associated Git commands. But first, let’s have a quick look at the organisation of Godot’s Git repository.

### 10.2.1 Git source repository

The repository on GitHub is a Git code repository together with an embedded issue tracker and PR system.
The Git version control system is the tool used to keep track of successive edits to the source code - to contribute efficiently to Godot, learning the basics of the Git command line is highly recommended. There exist some graphical interfaces for Git, but they usually encourage users to take bad habits regarding the Git and PR workflow, and we therefore recommend not to use them (especially GitHub’s online editor).

See also:

The first sections of Git’s “Book” are a good introduction to the tool’s philosophy and the various commands you need to master in your daily workflow. You can read them online on the Git SCM website.

The branches on the Git repository are organized as follows:

- The master branch is where the development of the next major version (3.0, 3.1, 4.0, etc.) occurs. As a development branch, it can be unstable and is not meant for use in production. This is where PRs should be done in priority.

- The stable branches are named after their version, e.g. 2.0 and 2.1. They are used to backport bugfixes and enhancements from the master branch to the currently maintained stable release (e.g. 2.0.1 or 2.1.3). As a rule of thumb, the last stable branch is maintained until the next major version (e.g. the 2.0 branch was maintained until the release of Godot 2.1). If you want to make PRs against a maintained stable branch, you will have to check if your changes are also relevant for the master branch.

- There might be feature branches at time, usually meant to be merged into the master branch at some time.

### 10.2.2 Forking and cloning

The first step is to fork the godotengine/godot repository on GitHub. To do so, you will need to have a GitHub account and to be logged in. In the top right corner of the repository’s GitHub page, you should see the “Fork” button as shown below:

![GitHub fork button](image)

Click it, and after a while you should be redirected to your own fork of the Godot repo, with your GitHub username as namespace:

```
| Unwatch | 801 | Unstar | 8,127 | Fork | 2,005 |
```

You can then clone your fork, i.e. create a local copy of the online repository (in Git speak, the origin remote):

```
$ git clone https://github.com/USERNAME/godot
```

**Note:** In our examples, the “$” character denotes the command line prompt on typical UNIX shells. It is not part of the command and should not be typed.

After a little while, you should have a godot directory in your current working directory. Move into it (cd godot), and we will set up a useful reference:

```
$ git remote add upstream https://github.com/godotengine/godot
$ git fetch upstream
```

This will create a reference named upstream pointing to the original godotengine/godot repository. This will be useful when you want to pull new commits from its master branch to update your fork. You have another remote reference named origin, which points to your fork.
You only need to do the above steps once, as long as you keep that local `godot` folder (which you can move around if you want, the relevant metadata is hidden in its `.git` subfolder).

**Note:** *Branch it, pull it, code it, stage it, commit, push it, rebase it... technologic.*

This bad take on Daft Punk’s *Technologic* shows the general conception Git beginners have of its workflow: lots of strange commands to learn by copy and paste, hoping they will work as expected. And that’s actually not a bad way to learn, as long as you’re curious and don’t hesitate to question your search engine when lost, so we will give you the basic commands to know when working in Git.

In the following, we will assume that you want to implement a feature in Godot’s project manager, which is coded in the `editor/project_manager.cpp` file.

### 10.2.3 Branching

By default, the `git clone` should have put you on the `master` branch of your fork (`origin`). To start your own feature development, we will create a feature branch:

```bash
$ git branch better-project-manager
$ git checkout better-project-manager
```

This command is equivalent:

```bash
$ git checkout -b better-project-manager
```

If you want to go back to the `master` branch, you’d use:

```bash
$ git checkout master
```

You can see which branch you are currently on with the `git branch` command:

```bash
$ git branch
2.1
* better-project-manager
  master
```

### 10.2.4 Updating your branch

This would not be needed the first time, just after you forked the upstream repository. However, the next time you want to work on something, you will notice that your fork’s `master` is several commits behind the upstream `master` branch: pull requests from other contributors would have been merged in the meantime.

To ensure there won’t be conflicts between the feature you develop and the current upstream `master` branch, you will have to update your branch by `pulling` the upstream branch.

```bash
$ git pull upstream master
```

However, if you had local commits, this method will create a so-called “merge commit”, and you will soon hear from fellow contributors that those are not wanted in PRs. Then how to update the branch without creating a merge commit? You will have to use the `--rebase` option, so that your local commits are replayed on top of the updated upstream `master` branch. It will effectively modify the Git history of your branch, but that is for the greater good.
Then command that you should (almost) always use is there:

```bash
$ git pull --rebase upstream master
```

## 10.2.5 Making changes

You would then do your changes to our example's `editor/project_manager.cpp` file with your usual development environment (text editor, IDE, etc.).

By default, those changes are *unstaged*. The staging area is a layer between your working directory (where you make your modifications) and the local git repository (the commits and all the metadata in the `.git` folder). To bring changes from the working directory to the git repository, you need to *stage* them with the `git add` command, and then to commit them with the `git commit` command.

There are various commands you should know to review your current work, before staging it, while it is staged, and after it has been committed.

- `git diff` will show you the current unstaged changes, i.e. the differences between your working directory and the staging area.
- `git checkout -- <files>` will undo the unstaged changes to the given files.
- `git add <files>` will *stage* the changes on the listed files.
- `git diff --staged` will show the current staged changes, i.e. the differences between the staging area and the last commit.
- `git reset HEAD <files>` will *unstage* changes to the listed files.
- `git status` will show you what are the currently staged and unstaged modifications.
- `git commit` will commit the staged files. It will open a text editor (you can define the one you want to use with the `GIT_EDITOR` environment variable or the `core.editor` setting in your Git config) to let you write a commit log. You can use `git commit -m "Cool commit log"` to write the log directly.
- `git log` will show you the last commits of your current branch. If you did local commits, they should be shown at the top.
- `git show` will show you the changes of the last commit. You can also specify a commit hash to see the changes for that commit.

That’s a lot to memorise! Don’t worry, just check this cheat sheet when you need to make changes, and learn by doing.

Here’s how the shell history could look like on our example:

```bash
// It's nice to know where you're starting from
$ git log
// Do changes to the project manager
$ nano editor/project_manager.cpp
// Find an unrelated bug in Control and fix it
$ nano scene/gui/control.cpp
// Review changes
$ git status
$ git diff
// We'll do two commits for our unrelated changes,
// starting by the Control changes necessary for the PM enhancements
$ git add scene/gui/control.cpp
$ git commit -m "Fix handling of margins in Control"
// Check we did good
$ git log
$ git show
```
With this, we should have two new commits in our `better-project-manager` branch which were not in the `master` branch. They are still only local though, the remote fork does not know about them, nor does the upstream repo.

### 10.2.6 Pushing changes to a remote

That’s where `git push` will come into play. In Git, a commit is always done in the local repository (unlike Subversion where a commit will modify the remote repository directly). You need to `push` the new commits to a remote branch to share them with the world. The syntax for this is:

```bash
$ git push <remote> <local branch>[:<remote branch>]
```

The part about the remote branch can be omitted if you want it to have the same name as the local branch, which is our case in this example, so we will do:

```bash
$ git push origin better-project-manager
```

Git will ask you for your username and password, and the changes will be sent to your remote. If you check the fork’s page on GitHub, you should see a new branch with your added commits.

### 10.2.7 Issuing a pull request

When you load your fork’s branch on GitHub, you should see a line saying “This branch is 2 commits ahead of godotengine:master.” (and potentially some commits behind, if your `master` branch was out of sync with the upstream `master` branch.

On that line, there is a “Pull request” link. Clicking it will open a form that will let you issue a pull request on the godotengine/godot upstream repository. It should show you your two commits, and state “Able to merge”. If not (e.g. it has way more commits, or says there are merge conflicts), don’t create the PR, something went wrong. Go to IRC and ask for support :)

Use an explicit title for the PR and put the necessary details in the comment area. You can drag and drop screenshots, gifs or zipped projects if relevant, to showcase what your work implements. Click “Create a pull request”, and tadaa!

### 10.2.8 Modifying a pull request

While it is reviewed by other contributors, you will often need to make changes to your yet-unmerged PR, either because contributors requested them, or because you found issues yourself while testing.

The good news is that you can modify a pull request simply by acting on the branch you made the pull request from. You can e.g. make a new commit on that branch, push it to your fork, and the PR will be updated automatically:

### 10.2. Pull request workflow
That should do the trick, but...

### 10.2.9 Mastering the PR workflow: the rebase

On the situation outlined above, your fellow contributors with an OCD regarding the Git history might ask you to rebase your branch to squash or meld the last two commits together (i.e. the two related to the project manager), as the second commit basically fixes an issue in the first one.

Once the PR is merged, it is not relevant for a changelog reader that the PR author made mistakes; instead, we want to keep only commits that bring from one working state to another working state.

To squash those two commits together, we will have to rewrite history. Right, we have that power. You may read that it’s a bad practice, and it’s true when it comes to branches of the upstream repo. But in your fork, you can do whatever you want, and everything is allowed to get neat PRs :)

We will use the interactive rebase git rebase -i to do this. This command takes a commit hash as argument, and will let you modify all commits between that commit hash and the last one of the branch, the so-called HEAD. In our example, we want to act on the last two commits, so we will do:

```
// The HEAD-X syntax means X commits before HEAD
$ git rebase -i HEAD~2
```

This will open a text editor with:

```
pick 1b4aad7 Add a pretty banner to the project manager
pick e07077e Fix a typo in the banner's title
```

The editor will also show instructions regarding how you can act on those commits. In particular, it should tell you that “pick” means to use that commit (do nothing), and that “squash” and “fixup” can be used to meld the commit in its parent commit. The difference between “squash” and “fixup” is that “fixup” will discard the commit log from the squashed commit. In our example, we are not interested in keeping the log of the “Fix a typo” commit, so we use:

```
pick 1b4aad7 Add a pretty banner to the project manager
fixup e07077e Fix a typo in the banner's title
```

Upon saving and quitting the editor, the rebase will occur. The second commit will be melded into the first one, and git log and git show should now confirm that you have only one commit with the changes from both previous commits.

**Note:** You could have avoided this rebase by using git commit --amend when fixing the typo. This command will write the staged changes directly into the last commit (HEAD), instead of creating a new commit like we did in this example. So it is equivalent to what we did with a new commit and then a rebase to mark it as “fixup”.

But! You rewrote the history, and now your local and remote branches have diverged. Indeed, commit 1b4aad7 in the above example will have changed, and therefore got a new commit hash. If you try to push to your remote branch, it will raise an error:
This is a sane behaviour, Git will not let you push changes that would override remote content. But that’s actually what we want to do here, so we will have to force it:

```
$ git push --force origin better-project-manager
```

And tadaa! Git will happily replace your remote branch with what you had locally (so make sure that’s what you wanted, using `git log`). This will also update the PR accordingly.

### 10.3 Code style guidelines

When contributing to Godot’s source code, you will be expected to follow the style guidelines outlined below. Some of them are checked via the Continuous Integration process and reviewers will ask you to fix potential issues, so best setup your system as outlined below to ensure all your commits follow the guidelines.

#### 10.3.1 C++ and Objective-C

There are no written guidelines, but the code style agreed upon by the developers is enforced via the `clang-format` code beautifier, which takes care for you of all our conventions. To name a few:

- Indentation and alignment are both tab based (respectively one and two tabs)
- One space around math and assignments operators as well as after commas
- Pointer and reference operators are affixed to the variable identifier, not to the type name

The rules used by clang-format are outlined in the `.clang-format` file of the Godot repository.

As long as you ensure that your style matches the surrounding code and that you not introducing trailing whitespace or space-based indentation, you should be fine. If you plan to contribute regularly however, we strongly advise that you setup clang-format locally to check and automatically fix all your commits.

**Warning:** Godot’s code style should not be applied to thirdparty code, i.e. that is included in Godot’s source tree but was not written specifically for our project. Such code usually come from different upstream projects with their own style guides (or lack thereof), and don’t want to introduce differences that would make syncing with upstream repositories harder.

Thirdparty code is usually included in the `thirdparty/` folder and can thus easily be excluded from formatting scripts. For the rare cases where a thirdparty code snippet needs to be included directly within a Godot file, you can use `/* clang-format off */` and `/* clang-format on */` to tell clang-format to ignore a chunk of code.

#### Using clang-format locally

First of all, you will need to install clang-format. As of now, you need to use `clang-format 5.x` to be compatible with Godot’s format. The upcoming 6.x branch has not been tested yet and my cause inconsistencies; the previous 3.x
branch is incompatible with the style definitions and will error out.

Installation

Here’s how to install clang-format:

- Linux: It will usually be available out-of-the-box with the clang toolchain packaged by your distribution. If your distro version is not the required one, you can download a pre-compiled version from the LLVM website, or if you are on a Debian derivative, use the upstream repos.

- macOS and Windows: You can download precompiled binaries from the LLVM website. You may need to add the path to the binary’s folder to your system’s PATH environment variable to be able to call clang-format out of the box.

You then have different possibilities to apply clang-format to your changes:

Manual usage

You can apply clang-format manually one or more files with the following command:

```
clang-format -i -style=file <path/to/file(s)>
```

- `-i` means that the changes should be written directly to the file (by default clang-format would only output the fixed version to the terminal).
- `-style=file` tells clang-format to use the `.clang-format` file of Godot’s repository as a style guide. Note that the `file` part is the actual “file” word, not a path to a file.
- The path can point to several files, either one after the other or using wildcards like in a typical Unix shell. Be careful when globbing so that you don’t run clang-format on compiled objects (.o and .a files) that are in Godot’s tree. So better use `core/*/.{cpp,h}` than `core/*`.

Pre-commit hook

For ease of use, we provide a pre-commit hook for Git that will run clang-format automatically on all your commits to check them, and let you apply its changes in the final commit.

This “hook” is a script which can be found in `misc/hooks`, refer to that folder’s README.md for installation instructions.

If your clang-format is not in the PATH, you may have to edit the `pre-commit-clang-format` to point to the correct binary for it to work. The hook was tested on Linux and macOS, but should also work in the Git Shell on Windows.

IDE plugin

Most IDEs or code editors have beautifier plugins that can be configured to run clang-format automatically, for example each time you save a file.

Here is a non-exhaustive list of beautifier plugins for some IDEs:

- Qt Creator: Beautifier plugin
- Visual Studio Code: Clang-Format
- Visual Studio: ClangFormat
10.3.2 Java

For Godot’s Java code (mostly in platform/android), there is currently no style guide, so for now try to stay consistent with the existing code.

Once a style is decided upon, it could also be enforced via clang-format.

10.3.3 Python

Godot’s SCons buildsystem is written in Python 2, and various scripts included in the source tree are either in Python 2 or Python 3.

For those, we follow the PEP-8 style guide, this is however not as strongly enforced as for the C++ code. If you are so inclined, you can check and format your Python changes using autopep8.

10.4 Bug triage guidelines

This page describes the typical workflow of the bug triage team aka bugsquad when handling issues and pull requests on Godot’s GitHub repository. It is bound to evolve together with the bugsquad, so do not hesitate to propose modifications to the following guidelines.

10.4.1 Issues management

GitHub proposes various features to manage issues:

- Set one or several labels from a predefined list
- Set one milestone from a predefined list
- Keep track of the issue in the project dashboard
- Define one contributor as “assignee” among the Godot engine organization members

As the Godot engine organization on GitHub currently has a restricted number of contributors, we do not use assignees extensively for now. All contributors are welcome to take on any issue, if relevant after mentioning it on the issue ticket and/or discussing the best way to resolve it with other developers.

For the time being we do not use the project dashboard feature either.

As far as possible, we try to assign labels (and milestones, when relevant) to both issues and pull requests.

Labels

The following labels are currently defined in the Godot repository:

Categories:

- *Archived:* either a duplicate of another issue, or invalid. Such an issue would also be closed.
- *Bug:* describes something that is not working properly.
• **Confirmed**: has been confirmed by at least one other contributor than the bug reporter (typically for Bug reports). The purpose of this label is to let developers know which issues are still reproducible when they want to select what to work on. It is therefore a good practice to add in a comment on what platform and what version or commit of Godot the issue could be reproduced; if a developer looks at the issue one year later, the *Confirmed* label may not be relevant anymore.

• **Discussion**: the issue is not consensual and needs further discussion to define what exactly should be done to address the topic.

• **Documentation**: issue related to the documentation. Mainly to request enhancements in the API documentation. Issues related to the ReadTheDocs documentation should be filed on the godot-docs repository.

• **Enhancement**: describes a proposed enhancement to an existing functionality.

• **Feature proposal**: describes a wish for a new feature to be implemented.

• **Junior job**: the issue is assumed to be an easy one to fix, which makes it a great fit for junior contributors who need to become familiar with the code base.

• **Needs rebase**: the issue need a git rebase to be merged.

• **Needs testing**: the issue/pull request could not be completely tested and thus need further testing. This can mean that it needs to be tested on different hardware/software configurations or even that the steps to reproduce are not certain.

• **PR welcome**: you are welcome to work on this issue and propose a pull request. Note that this *doesn’t* mean you can’t work on bugs without this label.

• **Tracker**: issue used to track other issues (like all issues related to the plugin system).

• **Usability**: issue that directly impact user usability.

The categories are used for general triage of the issues. They can be combined in some way when relevant, e.g. an issue can be labelled *Enhancement* and *Usability* at the same time if it’s an issue to improve usability. Or *Feature proposal* and *Discussion* if it’s a non-consensual feature request, or one that is not precise enough to be worked on.

**Topics:**

• **Assetlib**: relates to issues with the asset library.

• **Audio**: relates to the audio features (low and high level).

• **Buildsystem**: relates to building issues, either linked to the SCons buildsystem or to compiler peculiarities.

• **Core**: anything related to the core engine. It might be further split later on as it’s a pretty big topic.

• **Drivers**: relates to issues with the drivers used by the engine.

• **Editor**: relates to issues in the editor (mainly UI).

• **GDNative**: relates to the GDNative module.

• **GDScript**: relates to GDScript.

• **Physics**: relates to the physics engine (2D/3D).

• **Plugin**: relates to problems encountered while writing plugins.

• **Porting**: relates to some specific platforms.

• **Rendering**: relates to the 2D and 3D rendering engines.

• **VisualScript**: relates to issues with the visual scripting language.

Issues would typically correspond to only one topic, though it’s not unthinkable to see issues that fit two bills. The general idea is that there will be specialized contributors teams behind all topics, so they can focus on the issues labelled with their team’s topic.
Platforms: Android, HTML5, iOS, Linux, OS X, Windows, Winrt

By default, it is assumed that a given issue applies to all platforms. If one of the platform labels is used, it is then exclusive and the previous assumption doesn’t stand anymore (so if it’s a bug on e.g. Android and Linux exclusively, select those two platforms).

Milestones

Milestones correspond to planned future versions of Godot for which there is an existing roadmap. Issues that fit in the said roadmap should be filed under the corresponding milestone; if they don’t correspond to any current roadmap, they should be left without milestone. As a rule of thumb, an issue corresponds to a given milestone if it concerns a feature that is new in the milestone, or a critical bug that can’t be accepted in any future stable release, or anything that Juan wants to work on right now :)

Contributors are free to pick issues regardless of their assigned milestone; if a fix is proposed for a bug that was not deemed urgent and thus without milestone, it would likely still be very welcome.

10.5 Documentation guidelines

This page describes the rules to follow if you want to contribute to Godot Engine by writing or reviewing documentation, or by translating existing documentation.

10.5.1 How to contribute

Creating or modifying documentation pages is mainly done via the godot-docs GitHub repository. The HTML (or PDF and EPUB) documentation is generated from the .rst files (reStructuredText markup language) in that repository. Modifying those pages in a pull request and getting it merged will trigger a rebuild of the online documentation.

See also:

For details on Git usage and the pull request workflow, please refer to the Pull request workflow page. Most of what it describes regarding the main godotengine/godot repository is also valid for the docs repository.

The README.md file contains all the information you need to get you started, please read it. In particular, it contains some tips and tricks and links to reference documentation about the reStructuredText markup language.

Warning: If you want to edit the API reference, please note that it should not be done in the godot-docs repository. Instead, you should edit the doc/base/classes.xml XML file of Godot’s main repository. This file is then later used to generate the in-editor documentation as well as the API reference of the online docs.

10.5.2 What makes good documentation?

Documentation should be well written in plain English, using well-formed sentences and various levels of sections and subsections. It should be clear and objective.

We differentiate tutorial pages from other documentation pages by these definitions:

- Tutorial: a page aiming at explaining how to use one or more concepts in the editor or scripts in order to achieve a specific goal with a learning purpose (e.g. “Making a simple 2d Pong game”, “Applying forces to an object”).
- Documentation: a page describing precisely one and only one concept at a time, if possible exhaustively (e.g. the list of methods of the Sprite class, or an overview of the input management in Godot).
You are free to write the kind of documentation you wish, as long as you respect the following rules (and the ones on the repo). In particular, you can contribute tutorials in the “Community” section of the docs, where they could be merged relatively easily, improved by other contributors and eventually moved to an “official” section if relevant.

10.5.3 Titles

Always begin pages with their title and a Sphinx reference name:

.. _doc_insert_your_title_here:

Insert your title here

The reference allows to link to this page using the :ref: format, e.g. :ref:`doc_insert_your_title_here` would link to the above example page (note the lack of leading underscore in the reference).

Also, avoid American CamelCase titles: title’s first word should begin with a capitalized letter, and every following word should not. Thus, this is a good example:

• Insert your title here

And this is a bad example:

• Insert Your Title Here

Only project, people and node class names should have capitalized first letter.

10.5.4 Translating existing pages

For the moment, only English documentation is provided. We want to propose localized documentation in the future, but this is not a priority for now. Indeed, since the English documentation is still evolving a lot as the community improves it and make it more professional, we prefer that translators do not spend too much time on it, as their translated documentation might quickly grow out of sync with the original.

When the English documentation is ready for translations, we will provide tools to simplify the work of translators by tracking changes to the English docs that they should translate on their end.

10.5.5 Licence

This documentation and every page it contains is published under the terms of the Creative Commons Attribution 3.0 license (CC-BY-3.0), with attribution to “Juan Linietsky, Ariel Manzur and the Godot community”.

By contributing to the documentation on the GitHub repository, you agree that your changes are distributed under this license.

10.6 Updating the class reference

Godot Engine provides an extensive panel of nodes and singletons that you can use with GDScript to develop your games. All those classes are listed and documented in the class reference, which is available both in the online documentation and offline from within the engine.

The class reference is however not 100% complete. Some methods, constants and signals have not been described yet, while others may have their implementation changed and thus need an update. Updating the class reference is
a tedious work and the responsibility of the community: if we all partake in the effort, we can fill in the blanks and ensure a good documentation level in no time!

**Important notes:**

- To coordinate the effort and have an overview of the current status, we use a collaborative pad. Please follow the instructions there to notify the other documentation writers about what you are working on.
- We aim at completely filling the class reference in English first. Once it nears 90-95% completion, we could start thinking about localisating in other languages.

### 10.6.1 Workflow for updating the class reference

The source file of the class reference is an XML file in the main Godot source repository on GitHub, doc/base/classes.xml. As of now, it is relatively heavy (more than 1 MB), so it can’t be edited online using GitHub’s text editor.

The workflow to update the class reference is therefore:

- Fork the upstream repository and clone your fork.
- Edit the `doc/base/classes.xml` file, commit your changes and push to your fork.
- Make a pull request on the upstream repository.

The following section details this workflow, and gives also additional information about how to synchronise the XML template with the current state of the source code, or what should be the formatting of the added text.

**Important:** The class reference is also available in the online documentation, which is hosted alongside the rest of the documentation in the godot-docs git repository. The rST files should however not be edited directly, they are generated via a script from the `doc/base/classes.xml` file described above.

### 10.6.2 Getting started with GitHub

This section describes step-by-step the typical workflow to fork the git repository, or update an existing local clone of your fork, and then prepare a pull request.

**Fork Godot Engine**

First of all, you need to fork the Godot Engine on your own GitHub repository.

You will then need to clone the master branch of Godot Engine in order to work on the most recent version of the engine, including all of its features.

```bash
git clone https://github.com/your_name/godot.git
```

Then, create a new git branch that will contain your changes.

```bash
git checkout -b classref-edit
```

The branch you just created is identical to current master branch of Godot Engine. It already contains a `doc/` folder, with the current state of the class reference. Note that you can set whatever name you want for the branch, `classref-edit` is just an example.
Keeping your local clone up-to-date

If you already have a local clone of your own fork, it might happen pretty fast that it will diverge with the upstream git repository. For example other contributors might have updated the class reference since you last worked on it, or your own commits might have been squashed together when merging your pull request, and thus appear as diverging from the upstream master branch.

To keep your local clone up-to-date, you should first add an upstream git remote to work with:

```
git remote add upstream https://github.com/godotengine/godot
```

You only need to run this once to define this remote. The following steps will have to be run each time you want to sync your branch to the state of the upstream repo:

```
git pull --rebase upstream/master
```

This command would reapply your local changes (if any) on top of the current state of the upstream branch, thus bringing you up-to-date with your own commits on top. In case you have local commits that should be discarded (e.g. if your previous pull request had 5 small commits that were all merged into one bigger commit in the upstream branch), you need to reset your branch:

```
git fetch upstream
git reset --hard upstream/master
```

**Warning:** The above command will reset your branch to the state of the upstream/master branch, i.e. it will discard all changes which are specific to your local branch. So make sure to run this *before* making new changes and not afterwards.

Alternatively, you can also keep your own master branch (origin/master) up-to-date and create new branches when wanting to commit changes to the class reference:

```
git checkout master
git branch --d my-previous-doc-branch
```

```
git pull --rebase upstream/master
```

```
git checkout -b my-new-doc-branch
```

In case of doubt, ask for help on our IRC channels, we have some git gurus there.

Updating the documentation template

When classes are modified in the source code, the documentation template might become outdated. To make sure that you are editing an up-to-date version, you first need to compile Godot (you can follow the *Introduction to the buildsystem* page), and then run the following command (assuming 64-bit Linux):

```
./bin/godot.x11.tools.64 --doctool doc/base/classes.xml
```

The doc/base/classes.xml should then be up-to-date with current Godot Engine features. You can then check what changed using the *git diff* command. If there are changes to other classes than the one you are planning to document, please commit those changes first before starting to edit the template:

```
git add doc/base/classes.xml
```

```
git commit -m "Sync classes reference template with current code base"
```

You are now ready to edit this file to add stuff.
**Note:** If this has been done recently by another contributor, you don’t forcefully need to go through these steps (unless you know that the class you plan to edit has been modified recently).

**Push and request a pull of your changes**

Once your modifications are finished, push your changes on your GitHub repository:

```bash
git add doc/base/classes.xml
git commit -m "Explain your modifications."
git push
```

When it’s done, you can ask for a Pull Request via the GitHub UI of your Godot fork.

### 10.6.3 Editing the doc/base/classes.xml file

This file is generated and updated by Godot Engine. It is used by the editor as base for the Help section.

You can edit this file using your favourite text editor. If you use a code editor, make sure that it won’t needlessly change the indentation behaviour (e.g. change all tabs to spaces).

**Formatting of the XML file**

Here is an example with the Node2D class:

```xml
<class name="Node2D" inherits="CanvasItem" category="Core">
  <brief_description>
  Base node for 2D system.
  </brief_description>
  <description>
  Base node for 2D system. Node2D contains a position, rotation and scale, which is used to position and animate. It can alternatively be used with a custom 2D transform ([Matrix32]). A tree of Node2Ds allows complex hierarchies for animation and positioning.
  </description>
  <methods>
    <method name="set_pos">
      <argument index="0" name="pos" type="Vector2">
      </argument>
      <description>
      Set the position of the 2d node.
      </description>
    </method>
    <method name="set_rot">
      <argument index="0" name="rot" type="float">
      </argument>
      <description>
      Set the rotation of the 2d node.
      </description>
    </method>
    <method name="set_scale">
      <argument index="0" name="scale" type="Vector2">
      </argument>
      <description>
      Set the scale of the 2d node.
      </description>
    </method>
  </methods>
</class>
```
<method name="get_pos" qualifiers="const">
  <return type="Vector2">
  </return>
  <description>
  Return the position of the 2D node.
  </description>
</method>

<method name="get_rot" qualifiers="const">
  <return type="float">
  </return>
  <description>
  Return the rotation of the 2D node.
  </description>
</method>

<method name="get_scale" qualifiers="const">
  <return type="Vector2">
  </return>
  <description>
  Return the scale of the 2D node.
  </description>
</method>

<method name="rotate">
  <argument index="0" name="degrees" type="float">
  </argument>
  <description>
  </description>
</method>

<method name="move_local_x">
  <argument index="0" name="delta" type="float">
  </argument>
  <argument index="1" name="scaled" type="bool" default="false">
  </argument>
  <description>
  </description>
</method>

<method name="move_local_y">
  <argument index="0" name="delta" type="float">
  </argument>
  <argument index="1" name="scaled" type="bool" default="false">
  </argument>
  <description>
  </description>
</method>

<method name="get_global_pos" qualifiers="const">
  <return type="Vector2">
  </return>
  <description>
  Return the global position of the 2D node.
  </description>
</method>

<method name="set_global_pos">
  <argument index="0" name="arg0" type="Vector2">
  </argument>
  <description>
  </description>
</method>

<method name="set_transform">
  <description>
  </description>
</method>
As you can see, some methods in this class have no description (i.e. there is no text between their marks). This can also happen for the description and brief_description of the class, but in our example they are already filled. Let’s edit the description of the rotate() method:

```
<method name="rotate">
  <argument index="0" name="degrees" type="float">
    <description>
      Rotates the node of a given number of degrees.
    </description>
  </argument>
</method>
```

That’s all!

You simply have to write any missing text between these marks:

- `<description>`
- `<brief_description>`
- `<constant>`
- `<member>`
- `<signal>`

Describe clearly and shortly what the method does, or what the constant, member variable or signal mean. You can include an example of use if needed. Try to use grammatically correct English, and check the other descriptions to get an impression of the writing style.

For setters/getters, the convention is to describe in depth what the method does in the setter, and to say only the minimal in the getter to avoid duplication of the contents.

**Tags available for improved formatting**

For more control over the formatting of the help, Godot’s XML documentation supports various BBcode-like tags which are interpreted by both the offline in-editor Help, as well as the online documentation (via the reST converter).
Those tags are listed below. See existing documentation entries for more examples of how to use them properly.

<table>
<thead>
<tr>
<th>Tag</th>
<th>Effect</th>
<th>Usage</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Class]</td>
<td>Link a class</td>
<td>Move the [Sprite]</td>
<td>Move the Sprite.</td>
</tr>
<tr>
<td>[method methodname]</td>
<td>Link a method of this class</td>
<td>See [method set_pos].</td>
<td>See set_pos.</td>
</tr>
</tbody>
</table>

The [codeblock] is meant to be used for pre-formatted code block, using spaces as indentation (tabs will be removed by the reST converter). For example:

```
[codeblock]
func _ready():
    var sprite = get_node("Sprite")
    print(sprite.get_pos())
[/codeblock]
```

Which would be rendered as:

```
func _ready():
    var sprite = get_node("Sprite")
    print(sprite.get_pos())
```

**I don’t know what this method does!**

Not a problem. Leave it behind for now, and don’t forget to notify the missing methods when you request a pull of your changes. Another editor will take care of it.

If you wonder what a method does, you can still have a look at its implementation in Godot Engine’s source code on GitHub. Also, if you have a doubt, feel free to ask on the Q&A website and on IRC (freenode, #godotengine).
11.1 3D

11.1.1 Mesh generation with heightmap and shaders

Introduction

This tutorial will help you to use Godot shaders to deform a plane mesh so it appears like a basic terrain. Remember that this solution has pros and cons.

Pros:

• Pretty easy to do.
• This approach allows computation of LOD terrains.
• The heightmap can be used in Godot to create a normal map.

Cons:

• The Vertex Shader can’t re-compute normals of the faces. Thus, if your mesh is not static, this method will not work with shaded materials.
• This tutorial uses a plane mesh imported from Blender to Godot Engine. Godot is able to create meshes as well.

See this tutorial as an introduction, not a method that you should employ in your games, except if you intend to do LOD. Otherwise, this is probably not the best way.

However, let’s first create a heightmap, or a 2D representation of the terrain. To do this, I’ll use GIMP, but you can use any image editor you like.

The heightmap

We will use a few functions of GIMP image editor to produce a simple heightmap. Start GIMP and create a square image of 512x512 pixels.
You are now in front of a new, blank, square image.

Then, use a filter to render some clouds on this new image.
Parameter this filter to whatever you want. A white pixel corresponds to the highest point of the heightmap, a black pixel corresponds to the lowest one. So, darker regions are valleys and brighter are mountains. If you want, you can check “tileable” to render a heightmap that can be cloned and tiled close together with another one. X and Y size don’t matter a lot as long as they are big enough to provide a decent ground. A value of 4.0 or 5.0 for both is nice. Click on the “New Seed” button to roll a dice and GIMP will create a new random heightmap. Once you are happy with the result, click “OK”.

You can continue to edit your image if you wish. For our example, let’s keep the heightmap as is, and let’s export it to a PNG file, say “heightmap.png”. Save it in your Godot project folder.

The plane mesh

Now, we will need a plane mesh to import in Godot. Let’s run Blender.
Remove the start cube mesh, then add a new plane to the scene.

Zoom a bit, then switch to Edit mode (Tab key) and in the Tools buttongroup at the left, hit “Subdivide” 5 or 6 times.
Your mesh is now subdivided, which means we added vertices to the plane mesh that we will later be able to move. Job’s not finished yet: in order to texture this mesh a proper UV map is necessary. Currently, the default UV map contains only the 4 corner vertices we had at the beginning. However, we now have more, and we want to be able to texture over the whole mesh correctly.

If all the vertices of your mesh are not selected, select them all (hit “A”). They must appear orange, not black. Then, in the Shading/UVs button group to the left, click the “Unwrap” button (or simply hit “U”) and select “Smart UV Project”. Keep the default options and hit “Ok”.

Now, we need to switch our view to “UV/Image editor”.
Select all the vertices again ("A") then in the UV menu, select “Export UV Layout”.

Export the layout as a PNG file. Name it “plane.png” and save it in your Godot project folder. Now, let’s export our mesh as an OBJ file. Top of the screen, click “File/Export/Wavefront (obj)”. Save your object as “plane.obj” in your Godot project folder.

**Shader magic**

Let’s now open Godot Editor.
Create a new project in the folder you previously created and name it what you want.

In our default scene (3D), create a root node “Spatial”. Next, import the mesh OBJ file. Click “Import”, choose “3D Mesh” and select your plane.obj file, set the target path as “/” (or wherever you want in your project folder).

I like to check “Normals” in the import pop-up so the import will also consider faces normals, which can be useful (even if we don’t use them in this tutorial). Your mesh is now displayed in the FileSystem in “res://”.

11.1. 3D
Create a MeshInstance node. In the Inspector, load the mesh we just imported. Select “plane.msh” and hit ok.

Great! Our plane is now rendered in the 3D view.
It is time to add some shader stuff. In the Inspector, in the “Material Override” line, add a “New ShaderMaterial”. Edit it by clicking the “>” button just right to it.

You have two ways to create a shader: by code (MaterialShader), or using a shader graph (MaterialShaderGraph). The second one is a bit more visual, but we will not cover it for now. Create a “New MaterialShader”.
Edit it by clicking the “>” button just right to it. The Shaders editor opens.

The Vertex tab is for the Vertex shader, and the Fragment tab is for the Fragment shader. No need to explain what both of them do, right? If so, head to the Shading language page. Else, let’s start with the Fragment shader. This one is used to texture the plane using an image. For this example, we will texture it with the heightmap image itself, so we’ll actually see mountains as brighter regions and canyons as darker regions. Use this code:

```cpp
uniform texture source;
uniform color col;
DIFFUSE = col.rgb * tex(source,UV).rgb;
```
This shader is very simple (it actually comes from the *Shading language* page). What it basically does is take 2 parameters that we have to provide from outside the shader (“uniform”):

- the texture file
- a color

Then, we multiply every pixel of the image given by `tex(source,UV).rgb` by the color defined `col` and we set it to DIFFUSE variable, which is the rendered color. Remember that the UV variable is a shader variable that returns the 2D position of the pixel in the texture image, according to the vertex we are currently dealing with. That is the use of the UV Layout we made before. The color `col` is actually not necessary to display the texture, but it is interesting to play and see how it does, right?

However, the plane is displayed black! This is because we didn’t set the texture file and the color to use.

In the Inspector, click the “Previous” button to get back to the ShaderMaterial. This is where you want to set the texture and the color. In “Source”, click “Load” and select the texture file “heightmap.png”. But the mesh is still black! This is because our Fragment shader multiplies each pixel value of the texture by the `col` parameter. However, this color is currently set to black (0,0,0), and as you know, 0*x = 0 ;). Just change the `col` parameter to another color to see your texture appear:
Good. Now, the Vertex Shader.

The Vertex Shader is the first shader to be executed by the pipeline. It deals with vertices. Click the “Vertex” tab to switch, and paste this code:

```glsl
uniform texture source;
uniform float height_range;
vec2 xz = SRC_VERTEX.xz;
float h = tex(source, UV).g * height_range;
VERTEX = vec3(xz.x, h, xz.y);
VERTEX = MODELVIEW_MATRIX * VERTEX;
```

This shader uses two “uniform” parameters. The `source` parameter is already set for the fragment shader. Thus, the same image will be used in this shader as the heightmap. The `height_range` parameter is a parameter that we will use to increase the height effect.

At line 3, we save the x and z position of the SRC_VERTEX, because we do not want them to change: the plane must remain square. Remember that Y axis corresponds to the “altitude”, which is the only one we want to change with the heightmap.

At line 4, we compute an `h` variable by multiplying the pixel value at the UV position and the `height_range`. As the heightmap is a greyscale image, all r, g and b channels contain the same value. I used `g`, but any of r, g and b have the same effect.

At line 5, we set the current vertex’ position at (xz.x, h, xz.y) position. Concerning xz.y remember that its type is “vec2”. Thus, its components are x and y. The y component simply contains the z position we set at line 3.

Finally, at line 6, we multiply the vertex by the model/view matrix in order to set its position according to camera position. If you try to comment this line, you’ll see that the mesh behaves weird as you move and rotate the camera.

That’s all good, but our plane remains flat. This is because the `height_range` value is 0. Increase this value to observe the mesh distort and take to form of the terrain we set before:
11.1.2 Working with 3D skeletons

Godot 3D skeleton support is currently quite rudimentary. The `Skeleton` node and class were designed mainly to support importing skeletal animations as a set of transformation matrices.

**Skeleton node**

Skeleton node can be directly added anywhere you want on scene. Usually mesh is a child of Skeleton, as it easier to manipulate this way, as Transforms within skeleton are relative to where Skeleton is. But you can specify Skeleton node in every MeshInstance.

Being obvious, Skeleton is intended to deform meshes, and consists of structures called “bones”. Each “bone” is represented as Transform, which is applied to a group of vertices within a mesh. You can directly control a group of vertices from Godot. For that please reference `MeshDataTool` class, method `set_vertex_bones`. This class is very powerful.

The “bones” are organized in hierarchy, every bone, except for root bone(s) have parent. Every bone have associated name you can use to refer to it (e.g. “root” or “hand.L”, etc.). Also bones are all numbered, these numbers are bone IDs. Bone parents are referred by their numbered IDs.

For the rest of the article we consider the following scene:

```plaintext
main (Spatial) - script is always here
== skel (Skeleton)
==== mesh (MeshInstance)
```

This scene is imported from Blender. It contains arm mesh with 2 bones - upperarm and lowerarm, with lowerarm parented to upperarm.
Skeleton class

You can view Godot internal help for descriptions of every function. Basically all operations on bones are done using their numeric ID. You can convert from name to numeric ID and vice versa.

To find number of bones in skeleton we use get_bone_count() function

```godot
extends Spatial
var skel

func _ready():
    skel = get_node("skel")
    var id = skel.find_bone("upperarm")
    print("bone id:", id)
    var parent = skel.get_bone_parent(id)
    print("bone parent id:", id)
```

to find ID for the bone, use find_bone() function

```godot
extends Spatial
var skel

func _ready():
    skel = get_node("skel")
    var id = skel.find_bone("upperarm")
    print("bone id:", id)
```

Now, we want to do something interesting with ID except for printing it. Also, we might need additional information - to find bone parents to complete chain, etc. This all is done with get/set_bone_* functions.

To find bone parent we use get_bone_parent(id) function

```godot
extends Spatial
var skel

func _ready():
    skel = get_node("skel")
    var id = skel.find_bone("upperarm")
    print("bone id:", id)
    var parent = skel.get_bone_parent(id)
    print("bone parent id:", id)
```

Bone transforms is the thing why we’re here at all. There are 3 kind of transforms - local, global, custom.

To find bone local Transform we use get_bone_pose(id) function

```godot
extends Spatial
var skel

func _ready():
    skel = get_node("skel")
    var id = skel.find_bone("upperarm")
    print("bone id:", id)
    var parent = skel.get_bone_parent(id)
    print("bone parent id:", id)
    var t = skel.get_bone_pose(id)
    print("bone transform: ", t)
```

So we see 3x4 matrix there, with first column of 1s. What can we do about that? It is a Transform, so we can do
everything we can do with Transform, basically translate, rotate and scale. Also we can multiply transforms to have complex transforms. Remember, “bones” in Godot are just Transforms over a group of vertices. Also we can copy Transforms of other objects there. So lets rotate our “upperarm” bone:

```godot
extends Spatial
var skel
var id

func _ready():
    skel = get_node("skel")
    id = skel.find_bone("upperarm")
    print("bone id:", id)
    var parent = skel.get_bone_parent(id)
    print("bone parent id:", id)
    var t = skel.get_bone_pose(id)
    print("bone transform: ", t)
    set_process(true)

func _process(dt):
    var t = skel.get_bone_pose(id)
    t = t.rotated(Vector3(0.0, 1.0, 0.0), 0.1 * dt)
    skel.set_bone_pose(id, t)
```

Now we can rotate individual bones. The same happens for scale and translate - try these on your own and see results.

What we used now was local pose. By default all bones are not modified. But this Transform tells us nothing about relationship between bones. This information is needed for quite a number of tasks. How can we get it? Here comes global transform:

To find bone global Transform we use get_bone_global_pose(id) function

We will find global Transform for lowerarm bone:

```godot
extends Spatial
var skel

func _ready():
    skel = get_node("skel")
    var id = skel.find_bone("lowerarm")
    print("bone id:", id)
    var parent = skel.get_bone_parent(id)
    print("bone parent id:", id)
    var t = skel.get_bone_pose(id)
    print("bone transform: ", t)
```

As you see, this transform is not zeroed. While being called global, it is actually relative to Skeleton origin. For root bone, origin is always at 0 if not modified. Lets print origin for our lowerarm bone:

```godot
extends Spatial
var skel

func _ready():
    skel = get_node("skel")
    var id = skel.find_bone("lowerarm")
    print("bone id:", id)
    var parent = skel.get_bone_parent(id)
    print("bone parent id:", id)
    var t = skel.get_bone_global_pose(id)
    print("bone transform: ", t)
    print("bone origin: ", t.origin)
```
You will see a number. What does this number mean? It is a rotation point of Transform. So it is base part of the bone. In Blender you can go to Pose mode and try there to rotate bones - they will rotate around their origin. But what about tip? We can’t know things like bone length, which we need for many things, without knowing tip location. For all bones in chain except for last one we can calculate tip location - it is simply a child bone origin. Yes, there are situations when this is not true, for non-connected bones. But that is OK for us for now, as it is not important regarding Transforms. But the leaf bone tip is nowhere to be found. Leaf bone is a bone without children. So you don’t have any information about its tip. But this is not a showstopper. You can overcome this by either adding extra bone to the chain or just calculating leaf bone length in Blender and store the value in your script.

Using 3D “bones” for mesh control

Now as you know basics we can apply these to make full FK-control of our arm (FK is forward-kinematics)

To fully control our arm we need the following parameters:

- Upperarm angle x, y, z
- Lowerarm angle x, y, z

All of these parameters can be set, incremented and decremented.

Create the following node tree:

```
main (Spatial) <- script is here  
+--arm (arm scene)
      + DirectionLight (DirectionLight)
      + Camera
```

Set up Camera so that arm is properly visible. Rotate DirectionLight so that arm is properly lit while in scene play mode.

Now we need to create new script under main:

First we setup parameters:

```
var lowerarm_angle = Vector3()
var upperarm_angle = Vector3()
```

Now we need to setup a way to change them. Let us use keys for that.

Please create 7 actions under project settings:

- **select_x** - bind to X key
- **select_y** - bind to Y key
- **select_z** - bind to Z key
- **select_upperarm** - bind to key 1
- **select_lowerarm** - bind to key 2
- **increment** - bind to key numpad +
- **decrement** - bind to key numpad -

So now we want to adjust the above parameters. Therefore we create code which does that:

```
func _ready():
    set_process(true)
    var bone = "upperarm"
    var coordinate = 0
```
```python
func _process(dt):
    if Input.is_action_pressed("select_x"):
        coordinate = 0
    elif Input.is_action_pressed("select_y"):
        coordinate = 1
    elif Input.is_action_pressed("select_z"):
        coordinate = 2
    elif Input.is_action_pressed("select_upperarm"):
        bone = "upperarm"
    elif Input.is_action_pressed("select_lowerarm"):
        bone = "lowerarm"
    elif Input.is_action_pressed("increment"):
        if bone == "lowerarm":
            lowerarm_angle[coordinate] += 1
        elif bone == "upperarm":
            upperarm_angle[coordinate] += 1

The full code for arm control is this:

```extend Spatial

# member variables here, example:
# var a=2
# var b="textvar"
var upperarm_angle = Vector3()
var lowerarm_angle = Vector3()
var skel

func _ready():
    skel = get_node("arm/Armature/Skeleton")
    set_process(true)
var bone = "upperarm"
var coordinate = 0
func set_bone_rot(bone, ang):
    var b = skel.find_bone(bone)
    var rest = skel.get_bone_rest(b)
    var newpose = rest.rotated(Vector3(1.0, 0.0, 0.0), ang.x)
    var newpose = newpose.rotated(Vector3(0.0, 1.0, 0.0), ang.y)
    var newpose = newpose.rotated(Vector3(0.0, 0.0, 1.0), ang.z)
    skel.set_bone_pose(b, newpose)

func _process(dt):
    if Input.is_action_pressed("select_x"):
        coordinate = 0
    elif Input.is_action_pressed("select_y"):
        coordinate = 1
    elif Input.is_action_pressed("select_z"):
        coordinate = 2
    elif Input.is_action_pressed("select_upperarm"):
        bone = "upperarm"
    elif Input.is_action_pressed("select_lowerarm"):
        bone = "lowerarm"
    elif Input.is_action_pressed("increment"):
        if bone == "lowerarm":
            lowerarm_angle[coordinate] += 1
        elif bone == "upperarm":
            upperarm_angle[coordinate] += 1
    elif Input.is_action_pressed("decrement"):
```python
    if bone == "lowerarm":
        lowerarm_angle[coordinate] -= 1
    elif bone == "upperarm":
        upperarm_angle[coordinate] -= 1
    set_bone_rot("lowerarm", lowerarm_angle)
    set_bone_rot("upperarm", upperarm_angle)
```

Pressing keys 1/2 select upperarm/lowerarm, select axis by pressing x, y, z, rotate using numpad “+”/“-”.

This way you fully control your arm in FK mode using 2 bones. You can add additional bones and/or improve “feel” of the interface by using coefficients for the change. I recommend you play with this example a lot before going to next part.

You can clone the demo code for this chapter using

```bash
git clone git@github.com:slapin/godot-skel3d.git
cd demo1
```

Or you can browse it using web-interface:

https://github.com/slapin/godot-skel3d

Using 3D “bones” to implement Inverse Kinematics

See Inverse kinematics.

Using 3D “bones” to implement ragdoll-like physics

TODO.

11.1.3 Inverse kinematics

This tutorial is a follow-up of Working with 3D skeletons.

Before continuing on, I’d recommend reading some theory, the simplest article I find is this:

http://freespace.virgin.net/hugo.elias/models/m_ik2.htm

Initial problem

Talking in Godot terminology, the task we want to solve here is position our 2 angles we talked about above so, that the tip of lowerarm bone is as close to target point, which is set by target Vector3() as possible using only rotations. This task is very calculation-intensive and never resolved by analytical equation solve. So, it is an underconstrained problem, which means there is unlimited number of solutions to the equation.
For easy calculation, for this chapter we consider target is also child of Skeleton. If it is not the case for your setup you can always reparent it in your script, as you will save on calculations if you do.

In the picture you see angles alpha and beta. In this case we don’t use poles and constraints, so we need to add our own. On the picture the angles are 2D angles living in plane which is defined by bone base, bone tip and target.

The rotation axis is easily calculated using cross-product of bone vector and target vector. The rotation in this case will be always in positive direction. If \( t \) is the Transform which we get from `get_bone_global_pose()` function, the bone vector is

\[ t.basis[2] \]

So we have all information here to execute our algorithm.

In game dev it is common to resolve this problem by iteratively closing to the desired location, adding/subtracting small numbers to the angles until the distance change achieved is less than some small error value. Sounds easy enough, but there are Godot problems we need to resolve there to achieve our goal.

- **How to find coordinates of the tip of the bone?**
- **How to find vector from bone base to target?**

For our goal (tip of the bone moved within area of target), we need to know where the tip of our IK bone is. As we don’t use a leaf bone as IK bone, we know the coordinate of the bone base is the tip of parent bone. All these calculations are quite dependent on the skeleton’s structure. You can use pre-calculated constants as well. You can add an extra bone for the tip of IK and calculate using that.
Implementation

We will just use exported variable for bone length to be easy.

```godot
export var IK_bone="lowerarm"
export var IK_bone_length=1.0
export var IK_error = 0.1
```

Now, we need to apply our transformations from IK bone to the base of chain. So we apply rotation to IK bone then move from our IK bone up to its parent, then apply rotation again, then move to the parent of current bone again, etc. So we need to limit our chain somewhat.

```godot
export var IK_limit = 2
```

For `_ready()` function:

```godot
var skel
func _ready():
    skel = get_node("arm/Armature/Skeleton")
    set_process(true)
```

Now we can write our chain-passing function:

```godot
func pass_chain():
    var b = skel.find_bone(IK_bone)
    var l = IK_limit
    while b >= 0 and l > 0:
        print("name: ", skel.get_bone_name(b))
        print("local transform: ", skel.get_bone_pose(b))
        print("global transform:", skel.get_bone_global_pose(b))
        b = skel.get_bone_parent(b)
        l = l - 1
```

And for the `_process()` function:

```godot
func _process(dt):
    pass_chain(dt)
```

Executing this script will just pass through bone chain printing bone transforms.
Now we need to actually work with target. The target should be placed somewhere accessible. Since “arm” is imported scene, we better place target node within our top level scene. But for us to work with target easily its Transform should be on the same level as Skeleton.

To cope with this problem we create “target” node under our scene root node and at script run we will reparent it copying global transform, which will achieve wanted effect.

Create new Spatial node under root node and rename it to “target”. Then modify _ready() function to look like this:

```plaintext
var skel
var target
func _ready():
    skel = get_node("arm/Armature/Skeleton")
    target = get_node("target")
    var ttrans = target.get_global_transform()
    remove_child(target)
    skel.add_child(target)
    target.set_global_transform(ttrans)
    set_process(true)
```
CHAPTER 12

Godot API

12.1 @GDScript

Category: Core

12.1.1 Brief Description

Built-in GDScript functions.

12.1.2 Member Functions

<table>
<thead>
<tr>
<th>Color</th>
<th>Color8 ( int r8, int g8, int b8, int a8 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color</td>
<td>ColorN ( String name, float alpha )</td>
</tr>
<tr>
<td>float</td>
<td>abs ( float s )</td>
</tr>
<tr>
<td>float</td>
<td>acos ( float s )</td>
</tr>
<tr>
<td>float</td>
<td>asin ( float s )</td>
</tr>
<tr>
<td>Nil</td>
<td>assert ( bool condition )</td>
</tr>
<tr>
<td>float</td>
<td>atan ( float s )</td>
</tr>
<tr>
<td>float</td>
<td>atan2 ( float x, float y )</td>
</tr>
<tr>
<td>Variant</td>
<td>bytes2var ( RawArray bytes )</td>
</tr>
<tr>
<td>float</td>
<td>ceil ( float s )</td>
</tr>
<tr>
<td>float</td>
<td>clamp ( float val, float min, float max )</td>
</tr>
<tr>
<td>Object</td>
<td>convert ( Variant what, int type )</td>
</tr>
<tr>
<td>float</td>
<td>cos ( float s )</td>
</tr>
<tr>
<td>float</td>
<td>cosh ( float s )</td>
</tr>
<tr>
<td>float</td>
<td>db2linear ( float db )</td>
</tr>
<tr>
<td>float</td>
<td>decimals ( float step )</td>
</tr>
<tr>
<td>float</td>
<td>dec2time ( float value, float amount, float step )</td>
</tr>
<tr>
<td>float</td>
<td>deg2rad ( float deg )</td>
</tr>
<tr>
<td>Object</td>
<td>dict2inst ( Dictionary dict )</td>
</tr>
<tr>
<td>float</td>
<td>ease ( float s, float curve )</td>
</tr>
<tr>
<td>float</td>
<td>exp ( float s )</td>
</tr>
<tr>
<td>float</td>
<td>floor ( float s )</td>
</tr>
<tr>
<td>float</td>
<td>fmod ( float x, float y )</td>
</tr>
<tr>
<td>float</td>
<td>fposmod ( float x, float y )</td>
</tr>
</tbody>
</table>

Continued on next page
<table>
<thead>
<tr>
<th>FuncRef</th>
<th>funcref ( Object instance, String funcname )</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>hash ( Variant var:Variant )</td>
</tr>
<tr>
<td>Dictionary</td>
<td>inst2dict ( Object inst )</td>
</tr>
<tr>
<td>Object</td>
<td>instance_from_id ( int instance_id )</td>
</tr>
<tr>
<td>float</td>
<td>is_inf ( float s )</td>
</tr>
<tr>
<td>float</td>
<td>is_nan ( float s )</td>
</tr>
<tr>
<td>float</td>
<td>lerp ( float from, float to, float weight )</td>
</tr>
<tr>
<td>float</td>
<td>linear2db ( float nrg )</td>
</tr>
<tr>
<td>Resource</td>
<td>load ( String path )</td>
</tr>
<tr>
<td>float</td>
<td>log ( float s )</td>
</tr>
<tr>
<td>float</td>
<td>max ( float a, float b )</td>
</tr>
<tr>
<td>float</td>
<td>min ( float a, float b )</td>
</tr>
<tr>
<td>int</td>
<td>nearest_po2 ( int val )</td>
</tr>
<tr>
<td>float</td>
<td>pow ( float x, float y )</td>
</tr>
<tr>
<td>Resource</td>
<td>preload ( String path )</td>
</tr>
<tr>
<td>Nil</td>
<td>print ( Variant what, Variant ... )</td>
</tr>
<tr>
<td>Nil</td>
<td>print_stack ( )</td>
</tr>
<tr>
<td>Nil</td>
<td>printerr ( Variant what, Variant ... )</td>
</tr>
<tr>
<td>Nil</td>
<td>printraw ( Variant what, Variant ... )</td>
</tr>
<tr>
<td>Nil</td>
<td>prints ( Variant what, Variant ... )</td>
</tr>
<tr>
<td>Nil</td>
<td>printf ( Variant what, Variant ... )</td>
</tr>
<tr>
<td>float</td>
<td>rad2deg ( float rad )</td>
</tr>
<tr>
<td>float</td>
<td>rand_range ( float from, float to )</td>
</tr>
<tr>
<td>Array</td>
<td>rand_seed ( int seed )</td>
</tr>
<tr>
<td>float</td>
<td>randf ( )</td>
</tr>
<tr>
<td>int</td>
<td>randi ( )</td>
</tr>
<tr>
<td>Nil</td>
<td>randomize ( )</td>
</tr>
<tr>
<td>Array</td>
<td>range ( Variant ... )</td>
</tr>
<tr>
<td>float</td>
<td>round ( float s )</td>
</tr>
<tr>
<td>Nil</td>
<td>seed ( int seed )</td>
</tr>
<tr>
<td>float</td>
<td>sign ( float s )</td>
</tr>
<tr>
<td>float</td>
<td>sin ( float s )</td>
</tr>
<tr>
<td>float</td>
<td>sinh ( float s )</td>
</tr>
<tr>
<td>float</td>
<td>sqrt ( float s )</td>
</tr>
<tr>
<td>float</td>
<td>stepify ( float s, float step )</td>
</tr>
<tr>
<td>String</td>
<td>str ( Variant what, Variant ... )</td>
</tr>
<tr>
<td>Variant</td>
<td>str2var ( String string )</td>
</tr>
<tr>
<td>float</td>
<td>tan ( float s )</td>
</tr>
<tr>
<td>float</td>
<td>tanh ( float s )</td>
</tr>
<tr>
<td>bool</td>
<td>type_exists ( String type )</td>
</tr>
<tr>
<td>int</td>
<td>typeof ( Variant what )</td>
</tr>
<tr>
<td>RawArray</td>
<td>var2bytes ( Variant var )</td>
</tr>
<tr>
<td>String</td>
<td>var2str ( Variant var )</td>
</tr>
<tr>
<td>WeakRef</td>
<td>weakref ( Object obj )</td>
</tr>
<tr>
<td>GDFunctionState</td>
<td>yield ( Object object, String signal )</td>
</tr>
</tbody>
</table>

### 12.1.3 Numeric Constants

- **PI = 3.141593** — Constant that represents how many times the diameter of a circumference fits around its perimeter.
12.1.4 Description

This contains the list of built-in gdscript functions. Mostly math functions and other utilities. Everything else is expanded by objects.

12.1.5 Member Function Description

- **Color Color8** (int r8, int g8, int b8, int a8)
  Make a color from red, green, blue and alpha. Arguments can range from 0 to 255.
  - **Color ColorN** (String name, float alpha)
- **float abs** (float s)
  Remove sign (works for integer and float).
  - **float acos** (float s)
  Arc-cosine.
  - **float asin** (float s)
  Arc-sine.
- **Nil assert** (bool condition)
  Assert that the condition is true. If the condition is false, generates an error.
  - **float atan** (float s)
  Arc-tangent.
  - **float atan2** (float x, float y)
  Arc-tangent that takes a 2D vector as argument, returns the full -pi to +pi range.
- **Variant bytes2var** (RawArray bytes)
  Decode a byte array back to a value.
  - **float ceil** (float s)
  Ceiling (rounds up to nearest integer).
  - **float clamp** (float val, float min, float max)
  Clamp both values to a range.
- **Object convert** (Variant what, int type)
  Convert from a type to another in the best way possible. The “type” parameter uses the enum TYPE_* in @Global Scope.
  - **float cos** (float s)
  Standard cosine function.
  - **float cosh** (float s)
  Hyperbolic cosine.
  - **float db2linear** (float db)
  Convert from decibels to linear energy (audio).
  - **float decimals** (float step)
Return the amount of decimals in the floating point value.

- \texttt{float \ \textit{dectime} ( float value, float amount, float step )}

Decreases time by a specified amount.

- \texttt{float \ \textit{deg2rad} ( float deg )}

Convert from degrees to radians.

- \texttt{Object \ \textit{dict2inst} ( Dictionary dict )}

Convert a previously converted instances to dictionary back into an instance. Useful for deserializing.

- \texttt{float \ \textit{ease} ( float s, float curve )}

Easing function, based on exponent. 0 is constant, 1 is linear, 0 to 1 is ease-in, 1+ is ease out. Negative values are in-out/out in.

- \texttt{float \ \textit{exp} ( float s )}

Exponential logarithm.

- \texttt{float \ \textit{floor} ( float s )}

Floor (rounds down to nearest integer).

- \texttt{float \ \textit{fmod} ( float x, float y )}

Module (remainder of x/y).

- \texttt{float \ \textit{fposmod} ( float x, float y )}

Module (remainder of x/y) that wraps equally in positive and negative.

- \texttt{FuncRef \ \textit{funcref} ( Object instance, String funcname )}

Return a reference to the specified function.

- \texttt{int \ \textit{hash} ( Variant var:Variant )}

Hash the variable passed and return an integer.

- \texttt{Dictionary \ \textit{inst2dict} ( Object inst )}

Convert a script class instance to a dictionary (useful for serializing).

- \texttt{Object \ \textit{instance_from_id} ( int instance_id )}

Get an object by its ID.

- \texttt{float \ \textit{is_inf} ( float s )}

- \texttt{float \ \textit{is_nan} ( float s )}

- \texttt{float \ \textit{lerp} ( float from, float to, float weight )}

Linear interpolates between two values by a normalized value.

- \texttt{float \ \textit{linear2db} ( float nrg )}

Convert from linear energy to decibels (audio).

- \texttt{Resource \ \textit{load} ( String path )}

Load a resource from the filesystem, pass a valid path as argument.

- \texttt{float \ \textit{log} ( float s )}

Natural logarithm.
• \texttt{float max ( float a, float b )}
Return the maximum of two values.

• \texttt{float min ( float a, float b )}
Return the minimum of two values.

• \texttt{int nearest\_po2 ( int val )}
Return the nearest larger power of 2 for an integer.

• \texttt{float pow ( float x, float y )}
Power function, x elevate to y.

• \texttt{Resource preload ( String path )}
Preload a resource from the filesystem. The resource is loaded during script parsing.

• \texttt{Nil print ( Variant what, Variant \ldots )}
Print one or more arguments to strings in the best way possible to a console line.

• \texttt{Nil print\_stack ( )}
Print a stack track at code location, only works when running with debugger turned on.

• \texttt{Nil printerr ( Variant what, Variant \ldots )}
Print one or more arguments to strings in the best way possible to standard error line.

• \texttt{Nil printraw ( Variant what, Variant \ldots )}
Print one or more arguments to strings in the best way possible to console. No newline is added at the end.

• \texttt{Nil prints ( Variant what, Variant \ldots )}
Print one or more arguments to the console with a space between each argument.

• \texttt{Nil printt ( Variant what, Variant \ldots )}
Print one or more arguments to the console with a tab between each argument.

• \texttt{float rad2deg ( float rad )}
Convert from radians to degrees.

• \texttt{float rand\_range ( float from, float to )}
Random range, any floating point value between ‘from’ and ‘to’

• \texttt{Array rand\_seed ( int seed )}
Random from seed, pass a seed and an array with both number and new seed is returned.

• \texttt{float randf ( )}
Random value (0 to 1 float).

• \texttt{int randi ( )}
Random 32 bits value (integer). To obtain a value from 0 to N, you can use remainder, like (for random from 0 to 19): randi() \% 20.

• \texttt{Nil randomize ( )}
Reset the seed of the random number generator with a new, different one.

• \texttt{Array range ( Variant \ldots )}
Return an array with the given range. Range can be 1 argument N (0 to N-1), two arguments (initial, final-1) or three arguments (initial, final-1, increment).

- **float round ( float s )**
  Round to nearest integer.

- **Nil seed ( int seed )**
  Set seed for the random number generator.

- **float sign ( float s )**
  Return sign (-1 or +1).

- **float sin ( float s )**
  Standard sine function.

- **float sinh ( float s )**
  Hyperbolic sine.

- **float sqrt ( float s )**
  Square root.

- **float stepify ( float s, float step )**
  Snap float value to a given step.

- **String str ( Variant what, Variant ... )**
  Convert one or more arguments to strings in the best way possible.

- **Variant str2var ( String string )**
  Convert a formatted string that was returned by var2str to the original value.

- **float tan ( float s )**
  Standard tangent function.

- **float tanh ( float s )**
  Hyperbolic tangent.

- **bool type_exists ( String type )**
  Return the internal type of the given Variant object, using the TYPE_* enum in @Global Scope.

- **RawArray var2bytes ( Variant var )**
  Encode a variable value to a byte array.

- **String var2str ( Variant var )**
  Convert a value to a formatted string that can later be parsed using str2var.

- **WeakRef weakref ( Object obj )**
  Return a weak reference to an object.

- **GDFunctionState yield ( Object object, String signal )**
  Stop the function execution and return the current state. Call GDFunctionState.resume on the state to resume execution. This invalidates the state.
Returns anything that was passed to the resume function call. If passed an object and a signal, the execution is resumed when the object’s signal is emitted.

12.2 @Global Scope

Category: Core

12.2.1 Brief Description

Global scope constants and variables.

12.2.2 Member Variables

- **AudioServer AS** - [AudioServer] singleton
- **AudioServer AudioServer** - [AudioServer] singleton
- **Geometry Geometry** - [Geometry] singleton
- **Globals Globals** - [Globals] singleton
- **IP IP** - [IP] singleton
- **Input Input** - [Input] singleton
- **InputMap InputMap** - [InputMap] singleton
- **Reference Marshalls** - [Marshalls] singleton
- **OS OS** - [OS] singleton
- **PhysicsServer PS** - [PhysicsServer] singleton
- **Physics2DServer PS2D** - [Physics2DServer] singleton
- **PathRemap PathRemap** - [PathRemap] singleton
- **Performance Performance** - [Performance] singleton
- **Physics2DServer Physics2DServer** - [Physics2DServer] singleton
- **PhysicsServer PhysicsServer** - [PhysicsServer] singleton
- **SpatialSoundServer SS** - [SpatialSoundServer] singleton
- **SpatialSound2DServer SS2D** - [SpatialSound2DServer] singleton
- **SpatialSound2DServer SpatialSound2DServer** - [SpatialSound2DServer] singleton
- **SpatialSoundServer SpatialSoundServer** - [SpatialSoundServer] singleton
- **TranslationServer TS** - [TranslationServer] singleton
- **TranslationServer TranslationServer** - [TranslationServer] singleton
- **VisualServer VS** - [VisualServer] singleton
12.2.3 Numeric Constants

- `MARGIN_LEFT` = 0 — Left margin, used usually for `Control` or `StyleBox` derived classes.
- `MARGIN_TOP` = 1 — Top margin, used usually for `Control` or `StyleBox` derived classes.
- `MARGIN_RIGHT` = 2 — Right margin, used usually for `Control` or `StyleBox` derived classes.
- `MARGIN_BOTTOM` = 3 — Bottom margin, used usually for `Control` or `StyleBox` derived classes.
- `VERTICAL` = 1 — General vertical alignment, used usually for `Separator`, `ScrollBar`, `Slider`, etc.
- `HORIZONTAL` = 0 — General horizontal alignment, used usually for `Separator`, `ScrollBar`, `Slider`, etc.
- `HALIGN_LEFT` = 0 — Horizontal left alignment, usually for text-derived classes.
- `HALIGN_CENTER` = 1 — Horizontal center alignment, usually for text-derived classes.
- `HALIGN_RIGHT` = 2 — Horizontal right alignment, usually for text-derived classes.
- `VALIGN_TOP` = 0 — Vertical top alignment, usually for text-derived classes.
- `VALIGN_CENTER` = 1 — Vertical center alignment, usually for text-derived classes.
- `VALIGN_BOTTOM` = 2 — Vertical bottom alignment, usually for text-derived classes.
- `SPKEY` = 16777216 — Scancodes with this bit applied are non printable.
- `KEY_ESCAPE` = 16777217 — Escape Key
- `KEY_TAB` = 16777218 — Tab Key
- `KEY_BACKTAB` = 16777219 — Shift-Tab Key
- `KEY_BACKSPACE` = 16777220 — Backspace Key
- `KEY_RETURN` = 16777221 — Return Key (On Main Keyboard)
- `KEY_ENTER` = 16777222 — Enter Key (On Numpad)
- `KEY_INSERT` = 16777223 — Insert Key
- `KEY_DELETE` = 16777224 — Delete Key
- `KEY_PAUSE` = 16777225 — Pause Key
- `KEY_PRINT` = 16777226 — Printscreen Key
- `KEY_SYSREQ` = 16777227
- `KEY_CLEAR` = 16777228
- `KEY_HOME` = 16777229 — Home Key
- `KEY_END` = 16777230 — End Key
- `KEY_LEFT` = 16777231 — Left Arrow Key
- `KEY_UP` = 16777232 — Up Arrow Key
- `KEY_RIGHT` = 16777233 — Right Arrow Key
- `KEY_DOWN` = 16777234 — Down Arrow Key
- `KEY_PAGEUP` = 16777235 — Pageup Key
- `KEY_PAGEDOWN` = 16777236 — Pagedown Key
- `KEY_SHIFT` = 16777237 — Shift Key
- `KEY_CONTROL` = 16777238 — Control Key
- `KEY_META = 16777239`
- `KEY_ALT = 16777240` — Alt Key
- `KEY_CAPSLOCK = 16777241` — Capslock Key
- `KEY_NUMLOCK = 16777242` — Numlock Key
- `KEY_SCROLLLOCK = 16777243` — Scrolllock Key
- `KEY_F1 = 16777244` — F1 Key
- `KEY_F2 = 16777245` — F2 Key
- `KEY_F3 = 16777246` — F3 Key
- `KEY_F4 = 16777247` — F4 Key
- `KEY_F5 = 16777248` — F5 Key
- `KEY_F6 = 16777249` — F6 Key
- `KEY_F7 = 16777250` — F7 Key
- `KEY_F8 = 16777251` — F8 Key
- `KEY_F9 = 16777252` — F9 Key
- `KEY_F10 = 16777253` — F10 Key
- `KEY_F11 = 16777254` — F11 Key
- `KEY_F12 = 16777255` — F12 Key
- `KEY_F13 = 16777256` — F13 Key
- `KEY_F14 = 16777257` — F14 Key
- `KEY_F15 = 16777258` — F15 Key
- `KEY_F16 = 16777259` — F16 Key
- `KEY_KP_ENTER = 16777344` — Enter Key on Numpad
- `KEY_KP_MULTIPLY = 16777345` — Multiply Key on Numpad
- `KEY_KP_DIVIDE = 16777346` — Divide Key on Numpad
- `KEY_KP_SUBTRACT = 16777347` — Subtract Key on Numpad
- `KEY_KP_PERIOD = 16777348` — Period Key on Numpad
- `KEY_KP_ADD = 16777349` — Add Key on Numpad
- `KEY_KP_0 = 16777350` — Number 0 on Numpad
- `KEY_KP_1 = 16777351` — Number 1 on Numpad
- `KEY_KP_2 = 16777352` — Number 2 on Numpad
- `KEY_KP_3 = 16777353` — Number 3 on Numpad
- `KEY_KP_4 = 16777354` — Number 4 on Numpad
- `KEY_KP_5 = 16777355` — Number 5 on Numpad
- `KEY_KP_6 = 16777356` — Number 6 on Numpad
- `KEY_KP_7 = 16777357` — Number 7 on Numpad
- `KEY_KP_8 = 16777358` — Number 8 on Numpad
- KEY_KP_9 = 16777359 — Number 9 on Numpad
- KEY_SUPER_L = 16777260 — Super Left key (windows key)
- KEY_SUPER_R = 16777261 — Super Left key (windows key)
- KEY_MENU = 16777262 — Context menu key
- KEY_HYPER_L = 16777263
- KEY_HYPER_R = 16777264
- KEY_HELP = 16777265 — Help key
- KEY_DIRECTION_L = 16777266
- KEY_DIRECTION_R = 16777267
- KEY_BACK = 16777280 — Back key
- KEY_FORWARD = 16777281 — Forward key
- KEY_STOP = 16777282 — Stop key
- KEY_REFRESH = 16777283 — Refresh key
- KEY_VOLUMEDOWN = 16777284 — Volume down key
- KEY_VOLUMEMUTE = 16777285 — Mute volume key
- KEY_VOLUMEUP = 16777286 — Volume up key
- KEY_BASSBOOST = 16777287
- KEY_BASSUP = 16777288
- KEY_BASSDOWN = 16777289
- KEY_TREBLEUP = 16777290
- KEY_TREBLEDOWN = 16777291
- KEY_MEDIAPLAY = 16777292 — Media play key
- KEY_MEDIASTOP = 16777293 — Media stop key
- KEY_MEDIAPREVIOUS = 16777294 — Previous song key
- KEY_MEDIANEXT = 16777295 — Next song key
- KEY_MEDIARECORD = 16777296 — Media record key
- KEY_HOMEPAGE = 16777297 — Home page key
- KEY_FAVORITES = 16777298 — Favorites key
- KEY_SEARCH = 16777299 — Search key
- KEY_STANDBY = 16777300
- KEY_OPENURL = 16777301
- KEY_LAUNCHMAIL = 16777302
- KEY_LAUNCHMEDIA = 16777303
- KEY_LAUNCH0 = 16777304
- KEY_LAUNCH1 = 16777305
- KEY_LAUNCH2 = 16777306
• KEY_LAUNCH3 = 16777307
• KEY_LAUNCH4 = 16777308
• KEY_LAUNCH5 = 16777309
• KEY_LAUNCH6 = 16777310
• KEY_LAUNCH7 = 16777311
• KEY_LAUNCH8 = 16777312
• KEY_LAUNCH9 = 16777313
• KEY_LAUNCHA = 16777314
• KEY_LAUNCHB = 16777315
• KEY_LAUNCHC = 16777316
• KEY_LAUNCHD = 16777317
• KEY_LAUNCHE = 16777318
• KEY_LAUNCHF = 16777319
• KEY_UNKNOWN = 33554431
• KEY_SPACE = 32 — Space Key
• KEY_EXCLAM = 33 — ! key
• KEY_QUOTEDBL = 34 — " key
• KEY_NUMBERSIGN = 35 — # key
• KEY_DOLLAR = 36 — $ key
• KEY_PERCENT = 37 — % key
• KEY_AMPERSAND = 38 — & key
• KEY_APOSTROPHE = 39 — ' key
• KEY_PARENLEFT = 40 — ( key
• KEY_PARENRIGHT = 41 — ) key
• KEY_ASTERISK = 42 — * key
• KEY_PLUS = 43 — + key
• KEY_COMMA = 44 — , key
• KEY_MINUS = 45 — - key
• KEY_PERIOD = 46 — . key
• KEY_SLASH = 47 — / key
• KEY_0 = 48 — Number 0
• KEY_1 = 49 — Number 1
• KEY_2 = 50 — Number 2
• KEY_3 = 51 — Number 3
• KEY_4 = 52 — Number 4
• KEY_5 = 53 — Number 5
• KEY_6 = 54 — Number 6
• KEY_7 = 55 — Number 7
• KEY_8 = 56 — Number 8
• KEY_9 = 57 — Number 9
• KEY_COLON = 58 — : key
• KEY_SEMICOLON = 59 — ; key
• KEY_LESS = 60 — Lower than key
• KEY_EQUAL = 61 — = key
• KEY_GREATER = 62 — Greater than key
• KEY_QUESTION = 63 — ? key
• KEY_AT = 64 — @ key
• KEY_A = 65 — A Key
• KEY_B = 66 — B Key
• KEY_C = 67 — C Key
• KEY_D = 68 — D Key
• KEY_E = 69 — E Key
• KEY_F = 70 — F Key
• KEY_G = 71 — G Key
• KEY_H = 72 — H Key
• KEY_I = 73 — I Key
• KEY_J = 74 — J Key
• KEY_K = 75 — K Key
• KEY_L = 76 — L Key
• KEY_M = 77 — M Key
• KEY_N = 78 — N Key
• KEY_O = 79 — O Key
• KEY_P = 80 — P Key
• KEY_Q = 81 — Q Key
• KEY_R = 82 — R Key
• KEY_S = 83 — S Key
• KEY_T = 84 — T Key
• KEY_U = 85 — U Key
• KEY_V = 86 — V Key
• KEY_W = 87 — W Key
• KEY_X = 88 — X Key
• KEY_Y = 89 — Y Key
• KEY_Z = 90 — Z Key
• KEY_BRACKETLEFT = 91 — [ key
• KEY_BACKSLASH = 92 — key
• KEY_BRACKETRIGHT = 93 — ] key
• KEY_ASCIICIRCUM = 94 — ^ key
• KEY_UNDERSCORE = 95 — _ key
• KEY_QUOTELEFT = 96
• KEY_BRACELEFT = 123 — { key
• KEY_BAR = 124 — | key
• KEY_BRACERIGHT = 125 — } key
• KEY_ASCIITILDE = 126 — ~ key
• KEY_NOBREAKSPACE = 160
• KEY_EXCLAMDOWN = 161
• KEY_CENT = 162 — ¢ key
• KEY_STERLING = 163
• KEY_CURRENCY = 164
• KEY_YEN = 165
• KEY_BROKENBAR = 166 — ¦ key
• KEY_SECTION = 167 — § key
• KEY_DIAERESIS = 168 — ¨ key
• KEY_COPYRIGHT = 169 — © key
• KEY_ORDFEMININE = 170
• KEY_GUILLEMTLEFT = 171 — « key
• KEY_NOTSIGN = 172 — » key
• KEY_HYPHEN = 173 — - key
• KEY_REGISTERED = 174 — ® key
• KEY_MACRON = 175
• KEY_DEGREE = 176 — ° key
• KEY_PLUSMINUS = 177 — ± key
• KEY_TWOSUPERIOR = 178 — ² key
• KEY_THREESUPERIOR = 179 — ³ key
• KEY_ACUTE = 180 — ´ key
• KEY_MU = 181 — µ key
• KEY_PARAGRAPH = 182
• KEY_PERIODCENTERED = 183 — · key
• KEY_CEDILLA = 184 — ¬ key
- KEY_ONESUPERIOR = 185
- KEY_MASCULINE = 186
- KEY_GUILLEMETRIGHT = 187
- KEY_ONEQUARTER = 188
- KEY_ONEHALF = 189 — ½ key
- KEY_THREEQUARTERS = 190
- KEY_QUESTIONDOWN = 191
- KEY_AGRAVE = 192
- KEY_AACUTE = 193
- KEY_ACIRCUMFLEX = 194
- KEY_ATILDE = 195
- KEY_ADIACRISIS = 196
- KEY_ARING = 197
- KEY_AE = 198
- KEY_CCEDILLA = 199
- KEY_EGRAVE = 200
- KEY_EACUTE = 201
- KEY_ECIRCUMFLEX = 202
- KEY_EDIACRISIS = 203
- KEY_IGRAVE = 204
- KEY_IACUTE = 205
- KEY_ICIRCUMFLEX = 206
- KEY_IDIACRISIS = 207
- KEY_ETH = 208
- KEY_NTILDE = 209
- KEY_OGRAVE = 210
- KEY_OACUTE = 211
- KEY_OCIRCUMFLEX = 212
- KEY_OTILDE = 213
- KEY_ODIACRISIS = 214
- KEY_MULTIPLY = 215
- KEY_OOBLIQUE = 216
- KEY_UGRAVE = 217
- KEY_UACUTE = 218
- KEY_UCIRCUMFLEX = 219
- KEY_UDIACRISIS = 220
• KEY_YACUTE = 221
• KEY_THORN = 222
• KEY_SSHARP = 223
• KEY_DIVISION = 247
• KEY_YDIAERESIS = 255
• KEY_CODE_MASK = 33554431
• KEY_MODIFIER_MASK = -16777216
• KEY_MASK_SHIFT = 67108864
• KEY_MASK_ALT = 134217728
• KEY_MASK_CTRL = 268435456
• KEY_MASK_CMD = 268435456
• KEY_MASK_KPAD = 536870912
• KEY_MASK_GROUP_SWITCH = 1073741824
• BUTTON_LEFT = 1 — Left Mouse Button
• BUTTON_RIGHT = 2 — Right Mouse Button
• BUTTON_MIDDLE = 3 — Middle Mouse Button
• BUTTON_WHEEL_UP = 4 — Mouse wheel up
• BUTTON_WHEEL_DOWN = 5 — Mouse wheel down
• BUTTON_WHEEL_LEFT = 6 — Mouse wheel left button
• BUTTON_WHEEL_RIGHT = 7 — Mouse wheel right button
• BUTTON_MASK_LEFT = 1
• BUTTON_MASK_RIGHT = 2
• BUTTON_MASK_MIDDLE = 4
• JOY_BUTTON_0 = 0 — Joystick Button 0
• JOY_BUTTON_1 = 1 — Joystick Button 1
• JOY_BUTTON_2 = 2 — Joystick Button 2
• JOY_BUTTON_3 = 3 — Joystick Button 3
• JOY_BUTTON_4 = 4 — Joystick Button 4
• JOY_BUTTON_5 = 5 — Joystick Button 5
• JOY_BUTTON_6 = 6 — Joystick Button 6
• JOY_BUTTON_7 = 7 — Joystick Button 7
• JOY_BUTTON_8 = 8 — Joystick Button 8
• JOY_BUTTON_9 = 9 — Joystick Button 9
• JOY_BUTTON_10 = 10 — Joystick Button 10
• JOY_BUTTON_11 = 11 — Joystick Button 11
• JOY_BUTTON_12 = 12 — Joystick Button 12
• JOY_BUTTON_13 = 13 — Joystick Button 13
• JOY_BUTTON_14 = 14 — Joystick Button 14
• JOY_BUTTON_15 = 15 — Joystick Button 15
• JOY_BUTTON_MAX = 16 — Joystick Button 16
• JOY_SNES_A = 1 — Super Nintendo Entertainment System controller A button
• JOY_SNES_B = 0 — Super Nintendo Entertainment System controller B button
• JOY_SNES_X = 3 — Super Nintendo Entertainment System controller X button
• JOY_SNES_Y = 2 — Super Nintendo Entertainment System controller Y button
• JOY_SONY_CIRCLE = 1 — DUALSHOCK circle button
• JOY_SONY_X = 0 — DUALSHOCK X button
• JOY_SONY_SQUARE = 2 — DUALSHOCK square button
• JOY_SONY_TRIANGLE = 3 — DUALSHOCK triangle button
• JOY_SEGA_B = 1 — SEGA controller B button
• JOY_SEGA_A = 0 — SEGA controller A button
• JOY_SEGA_X = 2 — SEGA controller X button
• JOY_SEGA_Y = 3 — SEGA controller Y button
• JOY_XBOX_B = 1 — XBOX controller B button
• JOY_XBOX_A = 0 — XBOX controller A button
• JOY_XBOX_X = 2 — XBOX controller X button
• JOY_XBOX_Y = 3 — XBOX controller Y button
• JOY_DS_A = 1
• JOY_DS_B = 0
• JOY_DS_X = 3
• JOY_DS_Y = 2
• JOY_SELECT = 10 — Joystick Button Select
• JOY_START = 11 — Joystick Button Start
• JOY_DPAD_UP = 12 — Joystick DPad Up
• JOY_DPAD_DOWN = 13 — Joystick DPad Down
• JOY_DPAD_LEFT = 14 — Joystick DPad Left
• JOY_DPAD_RIGHT = 15 — Joystick DPad Right
• JOY_L = 4 — Joystick Left Shoulder Button
• JOY_L2 = 6 — Joystick Left Trigger
• JOY_L3 = 8 — Joystick Left Stick Click
• JOY_R = 5 — Joystick Right Shoulder Button
• JOY_R2 = 7 — Joystick Right Trigger
• **JOY_R3 = 9** — Joystick Right Stick Click  
• **JOY_AXIS_0 = 0** — Joystick Left Stick Horizontal Axis  
• **JOY_AXIS_1 = 1** — Joystick Left Stick Vertical Axis  
• **JOY_AXIS_2 = 2** — Joystick Right Stick Horizontal Axis  
• **JOY_AXIS_3 = 3** — Joystick Right Stick Vertical Axis  
• **JOY_AXIS_4 = 4**  
• **JOY_AXIS_5 = 5**  
• **JOY_AXIS_6 = 6** — Joystick Left Trigger Analog Axis  
• **JOY_AXIS_7 = 7** — Joystick Right Trigger Analog Axis  
• **JOY_AXIS_MAX = 8**  
• **JOY_ANALOG_0_X = 0** — Joystick Left Stick Horizontal Axis  
• **JOY_ANALOG_0_Y = 1** — Joystick Left Stick Vertical Axis  
• **JOY_ANALOG_1_X = 2** — Joystick Right Stick Horizontal Axis  
• **JOY_ANALOG_1_Y = 3** — Joystick Right Stick Vertical Axis  
• **JOY_ANALOG_2_X = 4**  
• **JOY_ANALOG_2_Y = 5**  
• **JOY_ANALOG_L2 = 6**  
• **JOY_ANALOG_R2 = 7**  

• **OK = 0** — Functions that return Error return OK when everything went ok. Most functions don’t return error anyway and/or just print errors to stdout.  
• **FAILED = 1** — Generic fail return error.  
• **ERR_UNAVAILABLE = 2**  
• **ERR_UNCONFIGURED = 3**  
• **ERR_UNAUTHORIZED = 4**  
• **ERR_PARAMETER_RANGE_ERROR = 5**  
• **ERR_OUT_OF_MEMORY = 6**  
• **ERR_FILE_NOT_FOUND = 7**  
• **ERR_FILE_BAD_DRIVE = 8**  
• **ERR_FILE_BAD_PATH = 9**  
• **ERR_FILE_NO_PERMISSION = 10**  
• **ERR_FILE_ALREADY_IN_USE = 11**  
• **ERR_FILE_CANT_OPEN = 12**  
• **ERR_FILE_CANT_WRITE = 13**  
• **ERR_FILE_CANT_READ = 14**  
• **ERR_FILE_UNRECOGNIZED = 15**  
• **ERR_FILE_CORRUPT = 16**
• ERR_FILE_MISSING_DEPENDENCIES = 17
• ERR_FILE_EOF = 18
• ERR_CANT_OPEN = 19
• ERR_CANT_CREATE = 20
• ERR_PARSE_ERROR = 43
• ERROR_QUERY_FAILED = 21
• ERR_ALREADY_IN_USE = 22
• ERR_LOCKED = 23
• ERR_TIMEOUT = 24
• ERR_CANT_AQUIRE_RESOURCE = 28
• ERR_INVALID_DATA = 30
• ERR_INVALID_PARAMETER = 31
• ERR_ALREADY_EXISTS = 32
• ERR_DOES_NOT_EXIST = 33
• ERR_DATABASE_CANT_READ = 34
• ERR_DATABASE_CANT_WRITE = 35
• ERR_COMPILATION_FAILED = 36
• ERR_METHOD_NOT_FOUND = 37
• ERR_LINK_FAILED = 38
• ERR_SCRIPT_FAILED = 39
• ERR_CYCLIC_LINK = 40
• ERR_BUSY = 44
• ERR_HELP = 46
• ERR_BUG = 47
• ERR_WTF = 49
• PROPERTY_HINT_NONE = 0 — No hint for edited property.
• PROPERTY_HINT_RANGE = 1 — Hints that the string is a range, defined as “min,max” or “min,max,step”. This is valid for integers and floats.
• PROPERTY_HINT_EXP_RANGE = 2 — Hints that the string is an exponential range, defined as “min,max” or “min,max,step”. This is valid for integers and floats.
• PROPERTY_HINT_ENUM = 3 — Property hint for an enumerated value, like “Hello,Something,Else”. This is valid for integer, float and string properties.
• PROPERTY_HINT_EXP_EASING = 4
• PROPERTY_HINT_LENGTH = 5
• PROPERTY_HINT_KEY_ACCEL = 7
• PROPERTY_HINT_FLAGS = 8 — Property hint for a bitmask description, for bits 0,1,2,3 and 5 the hint would be like “Bit0,Bit1,Bit2,Bit3,,Bit5”. Valid only for integers.
• **PROPERTY_HINT_ALL_FLAGS** = 9 — Property hint for a bitmask description that covers all 32 bits. Valid only for integers.

• **PROPERTY_HINT_FILE** = 10 — String property is a file (so pop up a file dialog when edited). Hint string can be a set of wildcards like “*.doc”.

• **PROPERTY_HINT_DIR** = 11 — String property is a directory (so pop up a file dialog when edited).

• **PROPERTY_HINT_GLOBAL_FILE** = 12

• **PROPERTY_HINT_GLOBAL_DIR** = 13

• **PROPERTY_HINT_RESOURCE_TYPE** = 14 — String property is a resource, so open the resource popup menu when edited.

• **PROPERTY_HINT_MULTILINE_TEXT** = 15

• **PROPERTY_HINT_COLOR_NO_ALPHA** = 16

• **PROPERTY_HINT_IMAGE_COMPRESS_LOSSY** = 17

• **PROPERTY_HINT_IMAGE_COMPRESS_LOSSLESS** = 18

• **PROPERTY_USAGE_STORAGE** = 1 — Property will be used as storage (default).

• **PROPERTY_USAGE_EDITOR** = 2 — Property will be visible in editor (default).

• **PROPERTY_USAGE_NETWORK** = 4

• **PROPERTY_USAGE_EDITOR_HELPER** = 8

• **PROPERTY_USAGE_CHECKABLE** = 16

• **PROPERTY_USAGE_CHECKED** = 32

• **PROPERTY_USAGE_INTERNATIONALIZED** = 64

• **PROPERTY_USAGE_BUNDLE** = 128

• **PROPERTY_USAGE_CATEGORY** = 256

• **PROPERTY_USAGE_STORE_IF_NONZERO** = 512

• **PROPERTY_USAGE_STORE_IF_NONONE** = 1024

• **PROPERTY_USAGE_NO_INSTANCE_STATE** = 2048

• **PROPERTY_USAGE_RESTART_IF_CHANGED** = 4096

• **PROPERTY_USAGE_SCRIPT_VARIABLE** = 8192

• **PROPERTY_USAGE_DEFAULT** = 7 — Default usage (storage and editor).

• **PROPERTY_USAGE_DEFAULT_INTL** = 71

• **PROPERTY_USAGE_NOEDITOR** = 5

• **METHOD_FLAG_NORMAL** = 1

• **METHOD_FLAG_EDITOR** = 2

• **METHOD_FLAG_NOSCRIPT** = 4

• **METHOD_FLAG_CONST** = 8

• **METHOD_FLAG_REVERSE** = 16

• **METHOD_FLAG_VIRTUAL** = 32

• **METHOD_FLAG_FROM_SCRIPT** = 64
• METHOD_FLAGS_DEFAULT = 1
• TYPE_NIL = 0 — Variable is of type nil (only applied for null).
• TYPE_BOOL = 1 — Variable is of type bool.
• TYPE_INT = 2 — Variable is of type int.
• TYPE_REAL = 3 — Variable is of type float/real.
• TYPE_STRING = 4 — Variable is of type String.
• TYPE_VECTOR2 = 5 — Variable is of type Vector2.
• TYPE_RECT2 = 6 — Variable is of type Rect2.
• TYPE_VECTOR3 = 7 — Variable is of type Vector3.
• TYPE_MATRIX32 = 8 — Variable is of type Matrix32.
• TYPE_PLANE = 9 — Variable is of type Plane.
• TYPE_QUAT = 10 — Variable is of type Quat.
• TYPE_AABB = 11 — Variable is of type AABB.
• TYPE_MATRIX3 = 12 — Variable is of type Matrix3.
• TYPE_TRANSFORM = 13 — Variable is of type Transform.
• TYPE_COLOR = 14 — Variable is of type Color.
• TYPE_IMAGE = 15 — Variable is of type Image.
• TYPE_NODE_PATH = 16 — Variable is of type NodePath.
• TYPE_RID = 17 — Variable is of type RID.
• TYPE_OBJECT = 18 — Variable is of type Object.
• TYPE_INPUT_EVENT = 19 — Variable is of type InputEvent.
• TYPE_DICTIONARY = 20 — Variable is of type Dictionary.
• TYPE_ARRAY = 21 — Variable is of type Array.
• TYPE_RAW_ARRAY = 22
• TYPE_INT_ARRAY = 23
• TYPE_REAL_ARRAY = 24
• TYPE_STRING_ARRAY = 25
• TYPE_VECTOR2_ARRAY = 26
• TYPE_VECTOR3_ARRAY = 27
• TYPE_COLOR_ARRAY = 28
• TYPE_MAX = 29

12.2.4 Description

Global scope constants and variables. This is all that resides in the globals, constants regarding error codes, scan codes, property hints, etc. It’s not much.

Singletons are also documented here, since they can be accessed from anywhere.
12.3 AABB

Category: Built-In Types

12.3.1 Brief Description

Axis-Aligned Bounding Box.

12.3.2 Member Functions

<table>
<thead>
<tr>
<th>AABB (Vector3 pos, Vector3 size)</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool encloses (AABB with)</td>
</tr>
<tr>
<td>AABB expand (Vector3 to_point)</td>
</tr>
<tr>
<td>float get_area ()</td>
</tr>
<tr>
<td>Vector3 get_endpoint (int idx)</td>
</tr>
<tr>
<td>Vector3 get_longest_axis ()</td>
</tr>
<tr>
<td>int get_longest_axis_index ()</td>
</tr>
<tr>
<td>float get_longest_axis_size ()</td>
</tr>
<tr>
<td>Vector3 get_shortest_axis ()</td>
</tr>
<tr>
<td>int get_shortest_axis_index ()</td>
</tr>
<tr>
<td>float get_shortest_axis_size ()</td>
</tr>
<tr>
<td>Vector3 get_support (Vector3 dir)</td>
</tr>
<tr>
<td>AABB grow (float by)</td>
</tr>
<tr>
<td>bool has_no_area ()</td>
</tr>
<tr>
<td>bool has_no_surface ()</td>
</tr>
<tr>
<td>bool has_point (Vector3 point)</td>
</tr>
<tr>
<td>AABB intersection (AABB with)</td>
</tr>
<tr>
<td>bool intersects (AABB with)</td>
</tr>
<tr>
<td>bool intersects_plane (Plane plane)</td>
</tr>
<tr>
<td>bool intersects_segment (Vector3 from, Vector3 to)</td>
</tr>
<tr>
<td>AABB merge (AABB with)</td>
</tr>
</tbody>
</table>

12.3.3 Member Variables

- Vector3 end - Ending corner.
- Vector3 pos - Position (starting corner).
- Vector3 size - Size from position to end.

12.3.4 Description

AABB provides an 3D Axis-Aligned Bounding Box. It consists of a position, a size, and several utility functions. It is typically used for simple (fast) overlap tests.

12.3.5 Member Function Description

- AABB (Vector3 pos, Vector3 size)

Optional constructor, accepts position and size.
• `bool encloses (AABB with)`
Return true if this `AABB` completely encloses another one.

• `AABB expand (Vector3 to_point)`
Return this `AABB` expanded to include a given point.

• `float get_area ()`
Get the area of the `AABB`.

• `Vector3 get_endpoint (int idx)`
Get the position of the 8 endpoints of the `AABB` in space.

• `Vector3 get_longest_axis ()`
Return the normalized longest axis of the `AABB`.

• `int get_longest_axis_index ()`
Return the index of the longest axis of the `AABB` (according to `Vector3::AXIS` enum).

• `float get_longest_axis_size ()`
Return the scalar length of the longest axis of the `AABB`.

• `Vector3 get_shortest_axis ()`
Return the normalized shortest axis of the `AABB`.

• `int get_shortest_axis_index ()`
Return the index of the shortest axis of the `AABB` (according to `Vector3::AXIS` enum).

• `float get_shortest_axis_size ()`
Return the scalar length of the shortest axis of the `AABB`.

• `Vector3 get_support (Vector3 dir)`
Return the support point in a given direction. This is useful for collision detection algorithms.

• `AABB grow (float by)`
Return a copy of the `AABB` grown a given amount of units towards all the sides.

• `bool has_no_area ()`
Return true if the `AABB` is flat or empty.

• `bool has_no_surface ()`
Return true if the `AABB` is empty.

• `bool has_point (Vector3 point)`
Return true if the `AABB` contains a point.

• `AABB intersection (AABB with)`
Return the intersection between two `AABB`. An empty AABB (size 0,0,0) is returned on failure.

• `bool intersects (AABB with)`
Return true if the `AABB` overlaps with another.

• `bool intersects_plane (Plane plane)`
Return true if the `AABB` is at both sides of a plane.
• `bool intersects_segment(Vector3 from, Vector3 to)`

Return true if the AABB intersects the line segment between from and to.

• `AABB merge(AABB with)`

Combine this AABB with another, a larger one is returned that contains both.

12.4 AcceptDialog

Inherits: `WindowDialog < Popup < Control < CanvasItem < Node < Object`

Inherited By: `ConfirmationDialog`

Category: Core

12.4.1 Brief Description

Base dialog for user notification.

12.4.2 Member Functions

<table>
<thead>
<tr>
<th>Button</th>
<th>add_button(String text, bool right=false, String action=&quot;&quot;)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Button</td>
<td>add_cancel(String name)</td>
</tr>
<tr>
<td>bool</td>
<td>get_hide_on_ok() const</td>
</tr>
<tr>
<td>Object</td>
<td>get_label() const</td>
</tr>
<tr>
<td>Object</td>
<td>get_ok()</td>
</tr>
<tr>
<td>String</td>
<td>get_text() const</td>
</tr>
<tr>
<td>LineEdit</td>
<td>register_text_enter(Object line_edit)</td>
</tr>
<tr>
<td>void</td>
<td>set_hide_on_ok(bool enabled)</td>
</tr>
<tr>
<td>void</td>
<td>set_text(String text)</td>
</tr>
</tbody>
</table>

12.4.3 Signals

• `confirmed()`

Emitted when accepted.

• `custom_action(String action)`

Emitted with a custom button is added.

12.4.4 Description

This dialog is useful for small notifications to the user about an event. It can only be accepted or closed, with the same result.
12.4.5 Member Function Description

- **Button** add_button (String text, bool right=false, String action="\"")
  Add custom button to the dialog and return the created button.
  The button titled with text and the action will be passed to custom_action signal when it is pressed.

- **Button** add_cancel (String name)
  Add custom cancel button to the dialog and return the created button.

- **bool** get_hide_on_ok () const
  Return true if the dialog will be hidden when accepted (default true).

- **Object** get_label ()
  Return the label used for built-in text.

- **Object** get_ok ()
  Return the OK Button.

- **String** get_text () const
  Return the built-in label text.

- **LineEdit** register_text_enter (Object line_edit)
  Register a LineEdit in the dialog. When the enter key is pressed, the dialog will be accepted.

- **void** set_hide_on_ok (bool enabled)
  Set whether the dialog is hidden when accepted (default true).

- **void** set_text (String text)
  Set the built-in label text.

12.5 AnimatedSprite

**Inherits:** Node2D < CanvasItem < Node < Object

**Category:** Core

12.5.1 Brief Description

Sprite node that can use multiple textures for animation.
12.5.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td><code>get_animation()</code> const</td>
<td>Return the name of the current animation set to the node.</td>
</tr>
<tr>
<td>int</td>
<td><code>get_frame()</code> const</td>
<td>Return the visible frame index.</td>
</tr>
<tr>
<td>Color</td>
<td><code>get_modulate()</code> const</td>
<td>Return the color modulation for this sprite.</td>
</tr>
<tr>
<td>Vector2</td>
<td><code>get_offset()</code> const</td>
<td>Return the offset of the sprite in the node origin.</td>
</tr>
<tr>
<td>SpriteFrames</td>
<td><code>get_sprite_frames()</code> const</td>
<td>Return the喷雾的喷雾。</td>
</tr>
<tr>
<td>bool</td>
<td><code>is_centered()</code> const</td>
<td></td>
</tr>
<tr>
<td>bool</td>
<td><code>is_flipped_h()</code> const</td>
<td></td>
</tr>
<tr>
<td>bool</td>
<td><code>is_flipped_v()</code> const</td>
<td></td>
</tr>
<tr>
<td>bool</td>
<td><code>is_playing()</code> const</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td><code>play(String anim=&quot;&quot;)</code></td>
<td></td>
</tr>
<tr>
<td>void</td>
<td><code>set_animation(String animation)</code></td>
<td></td>
</tr>
<tr>
<td>void</td>
<td><code>set_centered(bool centered)</code></td>
<td></td>
</tr>
<tr>
<td>void</td>
<td><code>set_flip_h(bool flip_h)</code></td>
<td></td>
</tr>
<tr>
<td>void</td>
<td><code>set_flip_v(bool flip_v)</code></td>
<td></td>
</tr>
<tr>
<td>void</td>
<td><code>set_frame(int frame)</code></td>
<td></td>
</tr>
<tr>
<td>void</td>
<td><code>set_modulate(Color modulate)</code></td>
<td></td>
</tr>
<tr>
<td>void</td>
<td><code>set_offset(Vector2 offset)</code></td>
<td></td>
</tr>
<tr>
<td>void</td>
<td><code>set_sprite_frames(SpriteFrames sprite_frames)</code></td>
<td></td>
</tr>
<tr>
<td>void</td>
<td><code>stop()</code></td>
<td></td>
</tr>
</tbody>
</table>

12.5.3 Signals

- **finished()**
  
  Emitted when the animation is finished (when it plays the last frame). If the animation is looping, this signal is emitted everytime the last frame is drawn, before looping.

- **frame_changed()**
  
  Emitted when frame is changed.

12.5.4 Description

Sprite node that can use multiple textures for animation.

12.5.5 Member Function Description

- **String get_animation() const**
  
  Return the name of the current animation set to the node.

- **int get_frame() const**
  
  Return the visible frame index.

- **Color get_modulate() const**
  
  Return the color modulation for this sprite.

- **Vector2 get_offset() const**
  
  Return the offset of the sprite in the node origin.
Get the `SpriteFrames` resource, which contains all frames.

- `bool is_centered()` const
  
  Return true when centered. See `set_centered`.

- `bool is_flipped_h()` const
  
  Return true if sprite is flipped horizontally.

- `bool is_flipped_v()` const
  
  Return true if sprite is flipped vertically.

- `bool is_playing()` const
  
  Return true if an animation if currently being played.

- `void play(String anim="")`
  
  Play the animation set in parameter. If no parameter is provided, the current animation is played.

- `void set_animation(String animation)`
  
  Set the current animation of the node and reinit the frame counter of the animation.

- `void set_centered(bool centered)`
  
  When turned on, offset at (0,0) is the center of the sprite, when off, the top-left corner is.

- `void set_flip_h(bool flip_h)`
  
  If true, sprite is flipped horizontally.

- `void set_flip_v(bool flip_v)`
  
  If true, sprite is flipped vertically.

- `void set_frame(int frame)`
  
  Set the visible sprite frame index (from the list of frames inside the `SpriteFrames` resource).

- `void set_modulate(Color modulate)`
  
  Change the color modulation (multiplication) for this sprite.

- `void set_offset(Vector2 offset)`
  
  Set the offset of the sprite in the node origin. Position varies depending on whether it is centered or not.

- `void set_sprite_frames(SpriteFrames sprite_frames)`
  
  Set the `SpriteFrames` resource, which contains all frames.

- `void stop()`
  
  Stop the current animation (does not reset the frame counter).

### 12.6 AnimatedSprite3D

**Inherits:** `SpriteBase3D < GeometryInstance < VisualInstance < Spatial < Node < Object`

**Category:** Core
12.6.1 Brief Description

12.6.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td><code>get_animation()</code> const</td>
</tr>
<tr>
<td>int</td>
<td><code>get_frame()</code> const</td>
</tr>
<tr>
<td>SpriteFrames</td>
<td><code>get_sprite_frames()</code> const</td>
</tr>
<tr>
<td>bool</td>
<td><code>is_playing()</code> const</td>
</tr>
<tr>
<td>void</td>
<td><code>play(String anim=&quot;&quot;)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_animation(String animation)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_frame(int frame)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_sprite_frames(SpriteFrames sprite_frames)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>stop()</code></td>
</tr>
</tbody>
</table>

12.6.3 Signals

- `frame_changed()`
  
  Emitted when frame is changed.

12.6.4 Member Function Description

- `String get_animation()` const
  
  Return the name of the current animation set to the node.

- `int get_frame()` const
  
  Return the visible frame index.

- `SpriteFrames get_sprite_frames()` const
  
  Get the `SpriteFrames` resource, which contains all frames.

- `bool is_playing()` const
  
  Return true if an animation if currently being played.

- `void play(String anim="")`
  
  Play the animation set in parameter. If no parameter is provided, the current animation is played.

- `void set_animation(String animation)`
  
  Set the current animation of the node and reinitializes the frame counter of the animation.

- `void set_frame(int frame)`
  
  Set the visible sprite frame index (from the list of frames inside the `SpriteFrames` resource).

- `void set_sprite_frames(SpriteFrames sprite_frames)`
  
  Set the `SpriteFrames` resource, which contains all frames.

- `void stop()`
  
  Stop the current animation (does not reset the frame counter).
12.7 Animation

Inherits: Resource < Reference < Object

Category: Core

12.7.1 Brief Description

Contains data used to animate everything in the engine.

12.7.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>add_track ( int type, int at_pos=-1 )</td>
</tr>
<tr>
<td>void</td>
<td>clear ()</td>
</tr>
<tr>
<td>int</td>
<td>find_track ( NodePath path ) const</td>
</tr>
<tr>
<td>float</td>
<td>get_length () const</td>
</tr>
<tr>
<td>float</td>
<td>get_step () const</td>
</tr>
<tr>
<td>int</td>
<td>get_track_count () const</td>
</tr>
<tr>
<td>bool</td>
<td>has_loop () const</td>
</tr>
<tr>
<td>IntArray</td>
<td>method_track_get_key_indices ( int idx, float time_sec, float delta ) const</td>
</tr>
<tr>
<td>String</td>
<td>method_track_get_name ( int idx, int key_idx ) const</td>
</tr>
<tr>
<td>Array</td>
<td>method_track_get_params ( int idx, int key_idx ) const</td>
</tr>
<tr>
<td>void</td>
<td>remove_track ( int idx )</td>
</tr>
<tr>
<td>void</td>
<td>set_length ( float time_sec )</td>
</tr>
<tr>
<td>void</td>
<td>set_loop ( bool enabled )</td>
</tr>
<tr>
<td>void</td>
<td>set_step ( float size_sec )</td>
</tr>
<tr>
<td>int</td>
<td>track_find_key ( int idx, float time, bool exact=false ) const</td>
</tr>
<tr>
<td>int</td>
<td>track_get_interpolation_type ( int idx ) const</td>
</tr>
<tr>
<td>int</td>
<td>track_get_key_count ( int idx ) const</td>
</tr>
<tr>
<td>float</td>
<td>track_get_key_time ( int idx, int key_idx ) const</td>
</tr>
<tr>
<td>float</td>
<td>track_get_key_transition ( int idx, int key_idx ) const</td>
</tr>
<tr>
<td>void</td>
<td>track_get_key_value ( int idx, int key_idx ) const</td>
</tr>
<tr>
<td>NodePath</td>
<td>track_get_path ( int idx ) const</td>
</tr>
<tr>
<td>int</td>
<td>track_get_type ( int idx ) const</td>
</tr>
<tr>
<td>void</td>
<td>track_insert_key ( int idx, float time, Variant key, float transition=1 )</td>
</tr>
<tr>
<td>bool</td>
<td>track_is_imported ( int idx ) const</td>
</tr>
<tr>
<td>void</td>
<td>track_move_down ( int idx )</td>
</tr>
<tr>
<td>void</td>
<td>track_move_up ( int idx )</td>
</tr>
<tr>
<td>void</td>
<td>track_remove_key ( int idx, int key_idx )</td>
</tr>
<tr>
<td>void</td>
<td>track_remove_key_at_pos ( int idx, float pos )</td>
</tr>
<tr>
<td>void</td>
<td>track_set_imported ( int idx, bool imported )</td>
</tr>
<tr>
<td>void</td>
<td>track_set_interpolation_type ( int idx, int interpolation )</td>
</tr>
<tr>
<td>void</td>
<td>track_set_key_transition ( int idx, int key_idx, float transition )</td>
</tr>
<tr>
<td>void</td>
<td>track_set_key_value ( int idx, int key, Variant value )</td>
</tr>
<tr>
<td>void</td>
<td>track_set_path ( int idx, NodePath path )</td>
</tr>
<tr>
<td>int</td>
<td>transform_track_insert_key ( int idx, float time, Vector3 loc, Quat rot, Vector3 scale )</td>
</tr>
<tr>
<td>Array</td>
<td>transform_track_interpolate ( int idx, float time_sec ) const</td>
</tr>
<tr>
<td>IntArray</td>
<td>value_track_get_key_indices ( int idx, float time_sec, float delta ) const</td>
</tr>
<tr>
<td>int</td>
<td>value_track_get_update_mode ( int idx ) const</td>
</tr>
</tbody>
</table>
### 12.7.3 Numeric Constants

- **TYPE_VALUE** = 0 — Value tracks set values in node properties, but only those which can be Interpolated.
- **TYPE_TRANSFORM** = 1 — Transform tracks are used to change node local transforms or skeleton pose bones. Transitions are Interpolated.
- **TYPE_METHOD** = 2 — Method tracks call functions with given arguments per key.
- **INTERPOLATION_NEAREST** = 0 — No interpolation (nearest value).
- **INTERPOLATION_LINEAR** = 1 — Linear interpolation.
- **INTERPOLATION_CUBIC** = 2 — Cubic interpolation.
- **UPDATE_CONTINUOUS** = 0
- **UPDATE_DISCRETE** = 1
- **UPDATE_TRIGGER** = 2

### 12.7.4 Description

An Animation resource contains data used to animate everything in the engine. Animations are divided into tracks, and each track must be linked to a node. The state of that node can be changed through time, by adding timed keys (events) to the track.

Animations are just data containers, and must be added to odes such as an `AnimationPlayer` or `AnimationTreePlayer` to be played back.

### 12.7.5 Member Function Description

- **int add_track ( int type, int at_pos=-1 )**
  Add a track to the Animation. The track type must be specified as any of the values in the TYPE_* enumeration.
- **void clear ( )**
  Clear the animation (clear all tracks and reset all).
- **int find_track ( NodePath path ) const**
  Return the index of the specified track. If the track is not found, return -1.
- **float get_length ( ) const**
  Return the total length of the animation (in seconds).
- **float get_step ( ) const**
  Get the animation step value.
- **int get_track_count ( ) const**
  Return the amount of tracks in the animation.
- **bool has_loop ( ) const**
  Return whether the animation has the loop flag set.
• **IntArray method_track_get_key_indices** (int idx, float time_sec, float delta) const
  Return all the key indices of a method track, given a position and delta time.

• **String method_track_get_name** (int idx, int key_idx) const
  Return the method name of a method track.

• **Array method_track_get_params** (int idx, int key_idx) const
  Return the arguments values to be called on a method track for a given key in a given track.

• **void remove_track** (int idx)
  Remove a track by specifying the track index.

• **void set_length** (float time_sec)
  Set the total length of the animation (in seconds). Note that length is not delimited by the last key, as this one may be before or after the end to ensure correct interpolation and looping.

• **void set_loop** (bool enabled)
  Set a flag indicating that the animation must loop. This is uses for correct interpolation of animation cycles, and for hinting the player that it must restart the animation.

• **void set_step** (float size_sec)
  Set the animation step value.

  • **int track_find_key** (int idx, float time, bool exact=false) const
    Find the key index by time in a given track. Optionally, only find it if the exact time is given.

  • **int track_get_interpolation_type** (int idx) const
    Return the interpolation type of a given track, from the INTERPOLATION_* enum.

  • **int track_get_key_count** (int idx) const
    Return the amount of keys in a given track.

  • **float track_get_key_time** (int idx, int key_idx) const
    Return the time at which the key is located.

  • **float track_get_key_transition** (int idx, int key_idx) const
    Return the transition curve (easing) for a specific key (see built-in math function “ease”).

  • **void track_get_key_value** (int idx, int key_idx) const
    Return the value of a given key in a given track.

  • **NodePath track_get_path** (int idx) const
    Get the path of a track. For more information on the path format, see `track_set_path`.

  • **int track_get_type** (int idx) const
    Get the type of a track.

• **void track_insert_key** (int idx, float time, Variant key, float transition=1)
  Insert a generic key in a given track.

• **bool track_is_imported** (int idx) const
  Return true if the given track is imported. Else, return false.

• **void track_move_down** (int idx)
Move a track down.

• void track_move_up ( int idx )

Move a track up.

• void track_remove_key ( int idx, int key_idx )

Remove a key by index in a given track.

• void track_remove_key_at_pos ( int idx, float pos )

Remove a key by position (seconds) in a given track.

• void track_set_imported ( int idx, bool imported )

Set the given track as imported or not.

• void track_set_interpolation_type ( int idx, int interpolation )

Set the interpolation type of a given track, from the INTERPOLATION_* enum.

• void track_set_key_transition ( int idx, int key_idx, float transition )

Set the transition curve (easing) for a specific key (see built-in math function “ease”).

• void track_set_key_value ( int idx, int key, Variant value )

Set the value of an existing key.

• void track_set_path ( int idx, NodePath path )

Set the path of a track. Paths must be valid scene-tree paths to a node, and must be specified starting from the parent node of the node that will reproduce the animation. Tracks that control properties or bones must append their name after the path, separated by “:”. Example: “character/skeleton:ankle” or “character/mesh:transform/local”

• int transform_track_insert_key ( int idx, float time, Vector3 loc, Quat rot, Vector3 scale )

Insert a transform key for a transform track.

• Array transform_track_interpolate ( int idx, float time_sec ) const

Return the interpolated value of a transform track at a given time (in seconds). An array consisting of 3 elements: position (Vector3), rotation (Quat) and scale (Vector3).

• IntArray value_track_get_key_indices ( int idx, float time_sec, float delta ) const

Return all the key indices of a value track, given a position and delta time.

• int value_track_get_update_mode ( int idx ) const

• void value_track_set_update_mode ( int idx, int mode )

### 12.8 AnimationPlayer

**Inherits:** Node < Object

**Category:** Core

**12.8.1 Brief Description**

Container and player of Animation resources.
12.8.2 Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>add_animation(String name, Animation animation)</code></td>
<td></td>
</tr>
<tr>
<td><code>advance(float delta)</code></td>
<td></td>
</tr>
<tr>
<td><code>animation_get_next(String anim_from) const</code></td>
<td></td>
</tr>
<tr>
<td><code>animation_set_next(String anim_from, String anim_to)</code></td>
<td></td>
</tr>
<tr>
<td><code>clear_caches()</code></td>
<td></td>
</tr>
<tr>
<td><code>clear_queue()</code></td>
<td></td>
</tr>
<tr>
<td><code>find_animation(Animation animation) const</code></td>
<td></td>
</tr>
<tr>
<td><code>get_animation(String name) const</code></td>
<td></td>
</tr>
<tr>
<td><code>get_animation_list() const</code></td>
<td></td>
</tr>
<tr>
<td><code>get_animation_process_mode() const</code></td>
<td></td>
</tr>
<tr>
<td><code>get_autoplay() const</code></td>
<td></td>
</tr>
<tr>
<td><code>get_blend_time(String anim_from, String anim_to) const</code></td>
<td></td>
</tr>
<tr>
<td><code>get_current_animation() const</code></td>
<td></td>
</tr>
<tr>
<td><code>get_current_animation_length() const</code></td>
<td></td>
</tr>
<tr>
<td><code>get_current_animation_pos() const</code></td>
<td></td>
</tr>
<tr>
<td><code>get_default_blend_time() const</code></td>
<td></td>
</tr>
<tr>
<td><code>get_pos() const</code></td>
<td></td>
</tr>
<tr>
<td><code>get_root() const</code></td>
<td></td>
</tr>
<tr>
<td><code>get_speed() const</code></td>
<td></td>
</tr>
<tr>
<td><code>has_animation(String name) const</code></td>
<td></td>
</tr>
<tr>
<td><code>is_active() const</code></td>
<td></td>
</tr>
<tr>
<td><code>is_playing() const</code></td>
<td></td>
</tr>
<tr>
<td><code>play(String name=&quot;&quot;, float custom_blend=-1, float custom_speed=1, bool from_end=false)</code></td>
<td></td>
</tr>
<tr>
<td><code>play_backwards(String name=&quot;&quot;, float custom_blend=-1)</code></td>
<td></td>
</tr>
<tr>
<td><code>queue(String name)</code></td>
<td></td>
</tr>
<tr>
<td><code>remove_animation(String name)</code></td>
<td></td>
</tr>
<tr>
<td><code>rename_animation(String name, String newname)</code></td>
<td></td>
</tr>
<tr>
<td><code>seek(float pos_sec, bool update=false)</code></td>
<td></td>
</tr>
<tr>
<td><code>set_active(bool active)</code></td>
<td></td>
</tr>
<tr>
<td><code>set_animation_process_mode(int mode)</code></td>
<td></td>
</tr>
<tr>
<td><code>set_autoplay(String name)</code></td>
<td></td>
</tr>
<tr>
<td><code>set_blend_time(String anim_from, String anim_to, float sec)</code></td>
<td></td>
</tr>
<tr>
<td><code>set_current_animation(String anim)</code></td>
<td></td>
</tr>
<tr>
<td><code>set_default_blend_time(float sec)</code></td>
<td></td>
</tr>
<tr>
<td><code>set_root(NodePath path)</code></td>
<td></td>
</tr>
<tr>
<td><code>set_speed(float speed)</code></td>
<td></td>
</tr>
<tr>
<td><code>stop(bool reset=true)</code></td>
<td></td>
</tr>
<tr>
<td><code>stop_all()</code></td>
<td></td>
</tr>
</tbody>
</table>

12.8.3 Signals

- **animation_changed(String old_name, String new_name)**
  
  If the currently being played animation changes, this signal will notify of such change.

- **animation_started(String name)**
  
  Notifies when an animation starts playing.

- **finished()**
  
  Notifies when an animation finished playing.
12.8.4 Numeric Constants

- \texttt{ANIMATION\_PROCESS\_FIXED} = 0 — Process animation on fixed process. This is specially useful when animating kinematic bodies.
- \texttt{ANIMATION\_PROCESS\_IDLE} = 1 — Process animation on idle process.

12.8.5 Description

An animation player is used for general purpose playback of \textit{Animation} resources. It contains a dictionary of animations (referenced by name) and custom blend times between their transitions. Additionally, animations can be played and blended in different channels.

12.8.6 Member Function Description

- \texttt{int add\_animation ( String name, Animation animation )}
  Add an animation resource to the player, which will be later referenced by the “name” argument.
- \texttt{void advance ( float delta )}
  Used to skip ahead or skip back in an animation. Delta is the time in seconds to skip.
- \texttt{String animation\_get\_next ( String anim\_from ) const}
- \texttt{void animation\_set\_next ( String anim\_from, String anim\_to )}
- \texttt{void clear\_caches ( )}
  The animation player creates caches for faster access to the nodes it will animate. However, if a specific node is removed, it may not notice it, so clear\_caches will force the player to search for the nodes again.
- \texttt{void clear\_queue ( )}
  If animations are queued to play, clear them.
- \texttt{String find\_animation ( Animation animation ) const}
  Find an animation name by resource.
- \texttt{Animation get\_animation ( String name ) const}
  Get an \textit{Animation} resource by requesting a name.
- \texttt{StringArray get\_animation\_list ( ) const}
  Get the list of names of the animations stored in the player.
- \texttt{int get\_animation\_process\_mode ( ) const}
  Return the mode in which the animation player processes. See \texttt{set\_animation\_process\_mode}.
- \texttt{String get\_autoplay ( ) const}
  Return the name of the animation that will be automatically played when the scene is loaded.
- \texttt{float get\_blend\_time ( String anim\_from, String anim\_to ) const}
  Get the blend time between two animations, referenced by their names.
- \texttt{String get\_current\_animation ( ) const}
  Return the name of the animation being played.
- \texttt{float get\_current\_animation\_length ( ) const}
Get the length (in seconds) of the currently being played animation.

- `float get_current_animation_pos() const`

Get the position (in seconds) of the currently being played animation.

- `float get_default_blend_time() const`

Return the default blend time between animations.

- `float get_pos() const`

Return the playback position (in seconds) in an animation channel (or channel 0 if none is provided).

- `NodePath get_root() const`

Return path to root node (see `set_root`).

- `float get_speed() const`

Get the speed scaling ratio in a given animation channel (or channel 0 if none is provided). Default ratio is 1 (no scaling).

- `bool has_animation(String name) const`

Request whether an Animation name exist within the player.

- `bool is_active() const`

Return true if the player is active.

- `bool is_playing() const`

Return whether an animation is playing.

- `void play(String name="", float custom_blend=-1, float custom_speed=1, bool from_end=false)`

Play a given animation by the animation name. Custom speed and blend times can be set. If custom speed is negative (-1), ‘from_end’ being true can play the animation backwards.

- `void play_backwards(String name="", float custom_blend=-1)`

Play a given animation by the animation name in reverse.

- `void queue(String name)`

Queue an animation for playback once the current one is done.

- `void remove_animation(String name)`

Remove an animation from the player (by supplying the same name used to add it).

- `void rename_animation(String name, String newname)`

Rename an existing animation.

- `void seek(float pos_sec, bool update=false)`

Seek the animation to a given position in time (in seconds). If ‘update’ is true, the animation will be updated too, otherwise it will be updated at process time.

- `void set_active(bool active)`

Set the player as active (playing). If false, it will do nothing.

- `void set_animation_process_mode(int mode)`

Set the mode in which the animation player processes. By default, it processes on idle time (framerate dependent), but using fixed time works well for animating static collision bodies in 2D and 3D. See enum ANIMATION_PROCESS_*. 
• **void set_autoplay ( String name )**

Set the name of the animation that will be automatically played when the scene is loaded.

• **void set_blend_time ( String anim_from, String anim_to, float sec )**

Specify a blend time (in seconds) between two animations, referenced by their names.

• **void set_current_animation ( String anim )**

Set the current animation (even if no playback occurs). Using set_current_animation() and set_active() are similar to calling play().

• **void set_default_blend_time ( float sec )**

Set the default blend time between animations.

• **void set_root ( NodePath path )**

AnimationPlayer resolves animation track paths from this node (which is relative to itself), by default root is ".\.", but it can be changed.

• **void set_speed ( float speed )**

Set a speed scaling ratio in a given animation channel (or channel 0 if none is provided). Default ratio is 1 (no scaling).

• **void stop ( bool reset=true )**

Stop the currently playing animation.

• **void stop_all ( )**

Stop playback of animations (deprecated).

### 12.9 AnimationTreePlayer

**Inherits:** Node < Object  
**Category:** Core

#### 12.9.1 Brief Description

Animation Player that uses a node graph for the blending.

#### 12.9.2 Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>void add_node ( int type, String id )</td>
<td></td>
</tr>
<tr>
<td>void advance ( float delta )</td>
<td></td>
</tr>
<tr>
<td>Animation animation_node_get_animation ( String id ) const</td>
<td></td>
</tr>
<tr>
<td>String animation_node_get_master_animation ( String id ) const</td>
<td></td>
</tr>
<tr>
<td>void animation_node_set_animation ( String id, Animation animation )</td>
<td></td>
</tr>
<tr>
<td>void animation_node_set_filter_path ( String id, NodePath path, bool enable )</td>
<td></td>
</tr>
<tr>
<td>void animation_node_set_master.Animation ( String id, String source )</td>
<td></td>
</tr>
<tr>
<td>float blend2_node_get_amount ( String id ) const</td>
<td></td>
</tr>
<tr>
<td>void blend2_node_set_amount ( String id, float blend )</td>
<td></td>
</tr>
</tbody>
</table>

Continued on next page
Table 12.4 – continued from previous page

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void blend2_node_set_filter_path(String id, NodePath path, bool enable)</td>
<td></td>
</tr>
<tr>
<td>float blend3_node_get_amount(String id) const</td>
<td></td>
</tr>
<tr>
<td>void blend3_node_set_amount(String id, float blend)</td>
<td></td>
</tr>
<tr>
<td>Vector2 blend4_node_get_amount(String id) const</td>
<td></td>
</tr>
<tr>
<td>void blend4_node_set_amount(String id, Vector2 blend)</td>
<td></td>
</tr>
<tr>
<td>int connect(String id, String dst_id, int dst_input_idx)</td>
<td></td>
</tr>
<tr>
<td>void disconnect(String id, int dst_input_idx)</td>
<td></td>
</tr>
<tr>
<td>int get_animation_process_mode() const</td>
<td></td>
</tr>
<tr>
<td>NodePath get_base_path() const</td>
<td></td>
</tr>
<tr>
<td>NodePath get_master_player() const</td>
<td></td>
</tr>
<tr>
<td>StringArray get_node_list()</td>
<td></td>
</tr>
<tr>
<td>bool is_active() const</td>
<td></td>
</tr>
<tr>
<td>bool is_connected(String id, String dst_id, int dst_input_idx) const</td>
<td></td>
</tr>
<tr>
<td>float mix_node_get_amount(String id) const</td>
<td></td>
</tr>
<tr>
<td>void mix_node_set_amount(String id, float ratio)</td>
<td></td>
</tr>
<tr>
<td>bool node_exists(String node) const</td>
<td></td>
</tr>
<tr>
<td>int node_get_input_count(String id) const</td>
<td></td>
</tr>
<tr>
<td>String node_get_input_source(String id, int idx) const</td>
<td></td>
</tr>
<tr>
<td>Vector2 node_get_pos(String id) const</td>
<td></td>
</tr>
<tr>
<td>int node_get_type(String id) const</td>
<td></td>
</tr>
<tr>
<td>int node_rename(String node, String new_name)</td>
<td></td>
</tr>
<tr>
<td>void node_set_pos(String id, Vector2 screen_pos)</td>
<td></td>
</tr>
<tr>
<td>float oneshot_node_get_autorestart_delay(String id) const</td>
<td></td>
</tr>
<tr>
<td>float oneshot_node_get_autorestart_random_delay(String id) const</td>
<td></td>
</tr>
<tr>
<td>float oneshot_node_get_fadein_time(String id) const</td>
<td></td>
</tr>
<tr>
<td>float oneshot_node_get_fadeout_time(String id) const</td>
<td></td>
</tr>
<tr>
<td>bool oneshot_node_has_autorestart(String id) const</td>
<td></td>
</tr>
<tr>
<td>bool oneshot_node_is_active(String id) const</td>
<td></td>
</tr>
<tr>
<td>void oneshot_node_set_autorestart(String id, bool enable)</td>
<td></td>
</tr>
<tr>
<td>void oneshot_node_set_autorestart_delay(String id, float delay_sec)</td>
<td></td>
</tr>
<tr>
<td>void oneshot_node_set_autorestart_random_delay(String id, float rand_sec)</td>
<td></td>
</tr>
<tr>
<td>void oneshot_node_set_fadein_time(String id, float time_sec)</td>
<td></td>
</tr>
<tr>
<td>void oneshot_node_set_fadeout_time(String id, float time_sec)</td>
<td></td>
</tr>
<tr>
<td>void oneshot_node_set_filter_path(String id, NodePath path, bool enable)</td>
<td></td>
</tr>
<tr>
<td>void oneshot_node_start(String id)</td>
<td></td>
</tr>
<tr>
<td>void oneshot_node_stop(String id)</td>
<td></td>
</tr>
<tr>
<td>void recompute_caches()</td>
<td></td>
</tr>
<tr>
<td>void remove_node(String id)</td>
<td></td>
</tr>
<tr>
<td>void reset()</td>
<td></td>
</tr>
<tr>
<td>void set_active(bool enabled)</td>
<td></td>
</tr>
<tr>
<td>void set_animation_process_mode(int mode)</td>
<td></td>
</tr>
<tr>
<td>void set_base_path(NodePath path)</td>
<td></td>
</tr>
<tr>
<td>void set_master_player(NodePath nodepath)</td>
<td></td>
</tr>
<tr>
<td>float timescale_node_get_scale(String id) const</td>
<td></td>
</tr>
<tr>
<td>void timescale_node_set_scale(String id, float scale)</td>
<td></td>
</tr>
<tr>
<td>void timeseek_node_seek(String id, float pos_sec)</td>
<td></td>
</tr>
<tr>
<td>void transition_node_delete_input(String id, int input_idx)</td>
<td></td>
</tr>
<tr>
<td>int transition_node_get_current(String id) const</td>
<td></td>
</tr>
<tr>
<td>int transition_node_get_input_count(String id) const</td>
<td></td>
</tr>
<tr>
<td>float transition_node_get_xfade_time(String id) const</td>
<td></td>
</tr>
</tbody>
</table>

Continued on next page
Table 12.4 – continued from previous page

<table>
<thead>
<tr>
<th>Type</th>
<th>Function Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool</td>
<td><code>transition_node_has_input_auto_advance (String id, int input_idx)</code> const</td>
</tr>
<tr>
<td>void</td>
<td><code>transition_node_set_current (String id, int input_idx)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>transition_node_set_input_auto_advance (String id, int input_idx, bool enable)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>transition_node_set_input_count (String id, int count)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>transition_node_set_xfade_time (String id, float time_sec)</code></td>
</tr>
</tbody>
</table>

12.9.3 Numeric Constants

- `NODE_OUTPUT` = 0
- `NODE_ANIMATION` = 1
- `NODE_ONESHOT` = 2
- `NODE_MIX` = 3
- `NODE_BLEND2` = 4
- `NODE_BLEND3` = 5
- `NODE_BLEND4` = 6
- `NODE_TIMESCALE` = 7
- `NODE_TIMESEEK` = 8
- `NODE_TRANSITION` = 9

12.9.4 Description

Animation Player that uses a node graph for the blending. This kind of player is very useful when animating character or other skeleton based rigs, because it can combine several animations to form a desired pose.

12.9.5 Member Function Description

- `void add_node (int type, String id)`
  Add a node of a given type in the graph with given id.
  - `void advance (float delta)`
  - `Animation animation_node_get_animation (String id) const`
  - `String animation_node_get_master_animation (String id) const`
  - `void animation_node_set_filter_path (String id, NodePath path, bool enable)`
  - `void animation_node_set_master_animation (String id, String source)`
  - `void animation_node_set_animation (String id, Animation animation)`
  
  Set the animation for an animation node.
  
  - `void animation_node_set_filter_path (String id, NodePath path, bool enable)`
  - `void animation_node_set_master_animation (String id, String source)`
  - `float blend2_node_get_amount (String id) const`
  - `void blend2_node_set_amount (String id, float blend)`
  - `void blend2_node_set_filter_path (String id, NodePath path, bool enable)`
  - `float blend3_node_get_amount (String id) const`
• **void** blend3_node_set_amount ( *String* id, *float* blend )

• **Vector2** blend4_node_get_amount ( *String* id ) const

• **void** blend4_node_set_amount ( *String* id, *Vector2* blend )

• **int** connect ( *String* id, *String* dst_id, *int* dst_input_idx )

• **void** disconnect ( *String* id, *int* dst_input_idx )

• **int** get_animation_process_mode ( ) const

• **NodePath** get_base_path ( ) const

• **NodePath** get_master_player ( ) const

• **StringArray** get_node_list ( )

• **bool** is_active ( ) const

• **bool** is_connected ( *String* id, *String* dst_id, *int* dst_input_idx ) const

• **float** mix_node_get_amount ( *String* id ) const

• **void** mix_node_set_amount ( *String* id, *float* ratio )

• **bool** node_exists ( *String* node ) const

• **int** node_get_input_count ( *String* id ) const

• **String** node_get_input_source ( *String* id, *int* idx ) const

• **Vector2** node_get_pos ( *String* id ) const

• **int** node_get_type ( *String* id ) const

• **NodePath** get_master_player ( ) const

Check if a node exists (by name).

• **int** node_get_input_count ( *String* id ) const

Return the input count for a given node. Different types of nodes have different amount of inputs.

• **String** node_get_input_source ( *String* id, *int* idx ) const

Return the input source for a given node input.

• **Vector2** node_get_pos ( *String* id ) const

• **int** node_get_type ( *String* id ) const

Get the node type, will return from NODE_* enum.

• **int** node_rename ( *String* node, *String* new_name )

Rename a node in the graph.

• **void** node_set_pos ( *String* id, *Vector2* screen_pos )

• **float** oneshot_node_get_autorestart_delay ( *String* id ) const

• **float** oneshot_node_get_autorestart_random_delay ( *String* id ) const

• **float** oneshot_node_get_fadein_time ( *String* id ) const

• **float** oneshot_node_get_fadeout_time ( *String* id ) const

• **bool** oneshot_node_has_autorestart ( *String* id ) const

• **bool** oneshot_node_is_active ( *String* id ) const

• **void** oneshot_node_set_autorestart ( *String* id, *bool* enable )

• **void** oneshot_node_set_autorestart_delay ( *String* id, *float* delay_sec )

• **void** oneshot_node_set_autorestart_random_delay ( *String* id, *float* rand_sec )

• **void** oneshot_node_set_fadein_time ( *String* id, *float* time_sec )

• **void** oneshot_node_set_fadeout_time ( *String* id, *float* time_sec )
- `void oneshot_node_set_filter_path ( String id, NodePath path, bool enable )`
- `void oneshot_node_start ( String id )`
- `void oneshot_node_stop ( String id )`
- `void recompute_caches ( )`
- `void remove_node ( String id )`
- `void reset ( )`
- `void set_active ( bool enabled )`
- `void set_animation_process_mode ( int mode )`
- `void set_base_path ( NodePath path )`
- `void set_master_player ( NodePath nodepath )`
- `float timescale_node_get_scale ( String id ) const`
- `void timescale_node_set_scale ( String id, float scale )`
- `void timeseek_node_seek ( String id, float pos_sec )`
- `void transition_node_delete_input ( String id, int input_idx )`
- `int transition_node_get_current ( String id ) const`
- `int transition_node_get_input_count ( String id ) const`
- `float transition_node_get_xfade_time ( String id ) const`
- `bool transition_node_has_input_auto_advance ( String id, int input_idx ) const`
- `void transition_node_set_current ( String id, int input_idx )`
- `void transition_node_set_input_auto_advance ( String id, int input_idx, bool enable )`
- `void transition_node_set_input_count ( String id, int count )`
- `void transition_node_set_xfade_time ( String id, float time_sec )`

### 12.10 Area

Inherits: `CollisionObject` < `Spatial` < `Node` < `Object`

Category: Core

#### 12.10.1 Brief Description

General purpose area detection and influence for 3D physics.

#### 12.10.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>float</td>
<td><code>get_angular_damp ( )</code> const</td>
</tr>
<tr>
<td>int</td>
<td><code>get_collision_mask ( )</code> const</td>
</tr>
<tr>
<td>bool</td>
<td><code>get_collision_mask_bit ( int bit )</code> const</td>
</tr>
</tbody>
</table>

Continued on next page
### 12.10.3 Signals

- **area_enter** *(Object area)*

  This signal is triggered only once when an area enters this area. The only parameter passed is the area that entered this area.

- **area_enter_shape** *(int area_id, Object area, int area_shape, int self_shape)*

  This signal triggers only once when an area enters this area. The first parameter is the area’s RID. The second one is the area as an object. The third one is the index of the shape entering this area, and the fourth one is the index of the shape in this area that reported the entering.

- **area_exit** *(Object area)*

  This signal is triggered only once when an area exits this area. The only parameter passed is the area that exited this area.

- **area_exit_shape** *(int area_id, Object area, int area_shape, int self_shape)*

  This signal triggers only once when an area exits this area. The first parameter is the area’s RID. The second one is the area as an object. The third one is the index of the shape entering this area, and the fourth one is the index of the shape in this area that reported the entering.

- **body_enter** *(Object body)*
This signal is triggered only once when a body enters this area. The only parameter passed is the body that entered this area.

- **body_enter_shape ( int body_id, Object body, int body_shape, int area_shape )**
  
  This signal triggers only once when a body enters this area. The first parameter is the body’s RID. The second one is the body as an object. The third one is the index of the shape of the body that entered this area, and the fourth one is the index of the shape in this area that reported the entering.

- **body_exit ( Object body )**
  
  This signal is triggered only once when a body exits this area. The only parameter passed is the body that exited this area.

- **body_exit_shape ( int body_id, Object body, int body_shape, int area_shape )**
  
  This signal triggers only once when a body exits this area. The first parameter is the body’s RID. The second one is the body as an object. The third one is the index of the shape exiting this area, and the fourth one is the index of the shape in this area that reported the exit.

### 12.10.4 Description

General purpose area detection for 3D physics. Areas can be used for detection of objects that enter/exit them, as well as overriding space parameters (changing gravity, damping, etc). For this, use any space override different from AREA_SPACE_OVERRIDE_DISABLE and point gravity at the center of mass.

### 12.10.5 Member Function Description

- **float get_angular_damp ( ) const**
  
  Return the angular damp rate.

- **int get_collision_mask ( ) const**
  
  Return the physics layers this area can scan for collisions.

- **bool get_collision_mask_bit ( int bit ) const**
  
  Return an individual bit on the collision mask.

- **float get_gravity ( ) const**
  
  Return the gravity intensity.

- **float get_gravity_distance_scale ( ) const**
  
  Return the falloff factor for point gravity.

- **Vector3 get_gravity_vector ( ) const**
  
  Return the gravity vector. If gravity is a point (see is_gravity_a_point), this will be the attraction center.

- **int get_layer_mask ( ) const**
  
  Return the physics layer this area is in.

- **bool get_layer_mask_bit ( int bit ) const**
  
  Return an individual bit on the layer mask.

- **float get_linear_damp ( ) const**
  
  Return the linear damp rate.
• **Array get_overlapping_areas ( ) const**

Return a list of the areas that are totally or partially inside this area.

• **Array get_overlapping_bodies ( ) const**

Return a list of the bodies (*PhysicsBody*) that are totally or partially inside this area.

• **float get_priority ( ) const**

Return the processing order of this area.

• **int get_space_override_mode ( ) const**

Return the space override mode.

• **bool is_gravity_a_point ( ) const**

Return whether gravity is a point. A point gravity will attract objects towards it, as opposed to a gravity vector, which moves them in a given direction.

• **bool is_monitorable ( ) const**

Return whether this area can be detected by other, monitoring, areas.

• **bool is_monitoring_enabled ( ) const**

Return whether this area detects bodies/areas entering/exiting it.

• **bool overlaps_area ( Object area ) const**

Return whether the area passed is totally or partially inside this area.

• **bool overlaps_body ( Object body ) const**

Return whether the body passed is totally or partially inside this area.

• **void set_angular_damp ( float angular_damp )**

Set the rate at which objects stop spinning in this area, if there are not any other forces making it spin. The value is a fraction of its current speed, lost per second. Thus, a value of 1.0 should mean stopping immediately, and 0.0 means the object never stops.

In practice, as the fraction of speed lost gets smaller with each frame, a value of 1.0 does not mean the object will stop in exactly one second. Only when the physics calculations are done at 1 frame per second, it does stop in a second.

• **void set_collision_mask ( int collision_mask )**

Set the physics layers this area can scan for collisions.

• **void set_collision_mask_bit ( int bit, bool value )**

Set/clear individual bits on the collision mask. This makes selecting the areas scanned easier.

• **void set_enable_monitoring ( bool enable )**

Set whether this area can detect bodies/areas entering/exiting it.

• **void set_gravity ( float gravity )**

Set the gravity intensity. This is useful to alter the force of gravity without altering its direction.

This value multiplies the gravity vector, whether it is the given vector (*set_gravity_vector*), or a calculated one (when using a center of gravity).

• **void set_gravity_distance_scale ( float distance_scale )**

Set the falloff factor for point gravity. The greater this value is, the faster the strength of gravity decreases with the square of distance.
• void `set_gravity_is_point(bool enable)`

When overriding space parameters, this method sets whether this area has a center of gravity. To set/get the location of the center of gravity, use `set_gravity_vector/get_gravity_vector`.

• void `set_gravity_vector(Vector3 vector)`

Set the gravity vector. This vector does not have to be normalized.

If gravity is a point (see `is_gravity_a_point`), this will be the attraction center.

• void `set_layer_mask(int layer_mask)`

Set the physics layers this area is in.

Collidable objects can exist in any of 32 different layers. These layers are not visual, but more of a tagging system instead. A collidable can use these layers/tags to select with which objects it can collide, using `set_collision_mask`.

A contact is detected if object A is in any of the layers that object B scans, or object B is in any layer scanned by object A.

• void `set_layer_mask_bit(int bit, bool value)`

Set/clear individual bits on the layer mask. This makes getting an area in/out of only one layer easier.

• void `set_linear_damp(float linear_damp)`

Set the rate at which objects stop moving in this area, if there are not any other forces moving it. The value is a fraction of its current speed, lost per second. Thus, a value of 1.0 should mean stopping immediately, and 0.0 means the object never stops.

In practice, as the fraction of speed lost gets smaller with each frame, a value of 1.0 does not mean the object will stop in exactly one second. Only when the physics calculations are done at 1 frame per second, it does stop in a second.

• void `set_monitorable(bool enable)`

Set whether this area can be detected by other, monitoring, areas. Only areas need to be marked as monitorable. Bodies are always so.

• void `set_priority(float priority)`

Set the order in which the area is processed. Greater values mean the area gets processed first. This is useful for areas which have an space override different from AREA_SPACE_OVERRIDE_DISABLED or AREA_SPACE_OVERRIDE_COMBINE, as they replace values, and are thus order-dependent.

Areas with the same priority value get evaluated in an unpredictable order, and should be differentiated if evaluation order is to be important.

• void `set_space_override_mode(int enable)`

Set the space override mode. This mode controls how an area affects gravity and damp.

AREA_SPACE_OVERRIDE_DISABLED: This area does not affect gravity/damp. These are generally areas that exist only to detect collisions, and objects entering or exiting them.

AREA_SPACE_OVERRIDE_COMBINE: This area adds its gravity/damp values to whatever has been calculated so far. This way, many overlapping areas can combine their physics to make interesting effects.

AREA_SPACE_OVERRIDE_COMBINE_REPLACE: This area adds its gravity/damp values to whatever has been calculated so far. Then stops taking into account the rest of the areas, even the default one.

AREA_SPACE_OVERRIDE_REPLACE: This area replaces any gravity/damp, even the default one, and stops taking into account the rest of the areas.

AREA_SPACE_OVERRIDE_REPLACE_COMBINE: This area replaces any gravity/damp calculated so far, but keeps calculating the rest of the areas, down to the default one.
12.11 Area2D

Inherits: `CollisionObject2D < Node2D < CanvasItem < Node < Object`

Category: Core

12.11.1 Brief Description

General purpose area detection and influence for 2D physics.

12.11.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>float</td>
<td><code>get_angular_damp()</code> const</td>
</tr>
<tr>
<td>int</td>
<td><code>get_collision_mask()</code> const</td>
</tr>
<tr>
<td>bool</td>
<td><code>get_collision_mask_bit(int bit)</code> const</td>
</tr>
<tr>
<td>float</td>
<td><code>get_gravity()</code> const</td>
</tr>
<tr>
<td>float</td>
<td><code>get_gravity_distance_scale()</code> const</td>
</tr>
<tr>
<td>Vector2</td>
<td><code>get_gravity_vector()</code> const</td>
</tr>
<tr>
<td>int</td>
<td><code>get_layer_mask()</code> const</td>
</tr>
<tr>
<td>bool</td>
<td><code>get_layer_mask_bit(int bit)</code> const</td>
</tr>
<tr>
<td>float</td>
<td><code>get_linear_damp()</code> const</td>
</tr>
<tr>
<td>Array</td>
<td><code>get_overlapping_areas()</code> const</td>
</tr>
<tr>
<td>Array</td>
<td><code>get_overlapping_bodies()</code> const</td>
</tr>
<tr>
<td>float</td>
<td><code>get_priority()</code> const</td>
</tr>
<tr>
<td>int</td>
<td><code>get_space_override_mode()</code> const</td>
</tr>
<tr>
<td>bool</td>
<td><code>is_gravity_a_point()</code> const</td>
</tr>
<tr>
<td>bool</td>
<td><code>is_monitorable()</code> const</td>
</tr>
<tr>
<td>bool</td>
<td><code>is_monitoring_enabled()</code> const</td>
</tr>
<tr>
<td>bool</td>
<td><code>overlaps_area(Object area)</code> const</td>
</tr>
<tr>
<td>bool</td>
<td><code>overlaps_body(Object body)</code> const</td>
</tr>
<tr>
<td>void</td>
<td><code>set_angular_damp(float angular_damp)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_collision_mask(int collision_mask)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_collision_mask_bit(int bit, bool)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_enable_monitoring(bool enable)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_gravity(float gravity)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_gravity_distance_scale(float distance_scale)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_gravity_is_point(bool enable)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_gravity_vector(Vector2 vector)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_layer_mask(int layer_mask)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_layer_mask_bit(int bit, bool value)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_linear_damp(float linear_damp)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_monitorable(bool enable)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_priority(float priority)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_space_override_mode(int enable)</code></td>
</tr>
</tbody>
</table>

12.11.3 Signals

- **area_enter(Object area)**

This signal is triggered only once when an area enters this area. The only parameter passed is the area that entered this
area.

- **area_enter_shape** (int area_id, Object area, int area_shape, int self_shape)
  This signal triggers only once when an area enters this area. The first parameter is the area’s RID. The second one is the area as an object. The third one is the index of the shape entering this area, and the fourth one is the index of the shape in this area that reported the entering.

- **area_exit** (Object area)
  This signal is triggered only once when an area exits this area. The only parameter passed is the area that exited this area.

- **area_exit_shape** (int area_id, Object area, int area_shape, int self_shape)
  This signal triggers only once when an area exits this area. The first parameter is the area’s RID. The second one is the area as an object. The third one is the index of the shape entering this area, and the fourth one is the index of the shape in this area that reported the entering.

- **body_enter** (Object body)
  This signal is triggered only once when a body enters this area. The only parameter passed is the body that entered this area.

- **body_enter_shape** (int body_id, Object body, int body_shape, int area_shape)
  This signal triggers only once when a body enters this area. The first parameter is the body’s RID. The second one is the body as an object. The third one is the index of the shape of the body that entered this area, and the fourth one is the index of the shape in this area that reported the entering.

- **body_exit** (Object body)
  This signal is triggered only once when a body exits this area. The only parameter passed is the body that exited this area.

- **body_exit_shape** (int body_id, Object body, int body_shape, int area_shape)
  This signal triggers only once when a body exits this area. The first parameter is the body’s RID. The second one is the body as an object. The third one is the index of the shape exiting this area, and the fourth one is the index of the shape in this area that reported the exit.

### 12.11.4 Description

General purpose area detection for 2D physics. Areas can be used for detection of objects that enter/exit them, as well as overriding space parameters (changing gravity, damping, etc). For this, use any space override different from AREA_SPACE_OVERRIDE_DISABLE and point gravity at the center of mass.

### 12.11.5 Member Function Description

- **float get_angular_damp** () const
  Return the angular damp rate.

- **int get_collision_mask** () const
  Return the physics layers this area can scan for collisions.

- **bool get_collision_mask_bit** (int bit) const
  Return an individual bit on the collision mask.

- **float get_gravity** () const
Return the gravity intensity.

- `float get_gravity_distance_scale ( ) const`

Return the falloff factor for point gravity.

- `Vector2 get_gravity_vector ( ) const`

Return the gravity vector. If gravity is a point (see `is_gravity_a_point`), this will be the attraction center.

- `int get_layer_mask ( ) const`

Return the physics layer this area is in.

- `bool get_layer_mask_bit ( int bit ) const`

Return an individual bit on the layer mask.

- `float get_linear_damp ( ) const`

Return the linear damp rate.

- `Array get_overlapping_areas ( ) const`

Return a list of the areas that are totally or partially inside this area.

- `Array get_overlapping_bodies ( ) const`

Return a list of the bodies (`PhysicsBody2D`) that are totally or partially inside this area.

- `float get_priority ( ) const`

Return the processing order of this area.

- `int get_space_override_mode ( ) const`

Return the space override mode.

- `bool is_gravity_a_point ( ) const`

Return whether gravity is a point. A point gravity will attract objects towards it, as opposed to a gravity vector, which moves them in a given direction.

- `bool is_monitorable ( ) const`

Return whether this area can be detected by other, monitoring, areas.

- `bool is_monitoring_enabled ( ) const`

Return whether this area detects bodies/areas entering/exiting it.

- `bool overlaps_area ( Object area ) const`

Return whether the area passed is totally or partially inside this area.

- `bool overlaps_body ( Object body ) const`

Return whether the body passed is totally or partially inside this area.

- `void set_angular_damp ( float angular_damp )`

Set the rate at which objects stop spinning in this area, if there are not any other forces making it spin. The value is a fraction of its current speed, lost per second. Thus, a value of 1.0 should mean stopping immediately, and 0.0 means the object never stops.

In practice, as the fraction of speed lost gets smaller with each frame, a value of 1.0 does not mean the object will stop in exactly one second. Only when the physics calculations are done at 1 frame per second, it does stop in a second.

- `void set_collision_mask ( int collision_mask )`
Set the physics layers this area can scan for collisions.

- **void set_collision_mask_bit ( int bit, bool value )**

Set/clear individual bits on the collision mask. This makes selecting the areas scanned easier.

- **void set_enable_monitoring ( bool enable )**

Set whether this area can detect bodies/areas entering/exiting it.

- **void set_gravity ( float gravity )**

Set the gravity intensity. This is useful to alter the force of gravity without altering its direction.

This value multiplies the gravity vector, whether it is the given vector (set_gravity_vector), or a calculated one (when using a center of gravity).

- **void set_gravity_distance_scale ( float distance_scale )**

Set the falloff factor for point gravity. The greater this value is, the faster the strength of gravity decreases with the square of distance.

- **void set_gravity_is_point ( bool enable )**

When overriding space parameters, this method sets whether this area has a center of gravity. To set/get the location of the center of gravity, use set_gravity_vector/get_gravity_vector.

- **void set_gravity_vector ( Vector2 vector )**

Set the gravity vector. This vector does not have to be normalized. If gravity is a point (see is_gravity_a_point), this will be the attraction center.

- **void set_layer_mask ( int layer_mask )**

Set the physics layers this area is in.

Collidable objects can exist in any of 32 different layers. These layers are not visual, but more of a tagging system instead. A collidable can use these layers/tags to select with which objects it can collide, using set_collision_mask.

A contact is detected if object A is in any of the layers that object B scans, or object B is in any layer scanned by object A.

- **void set_layer_mask_bit ( int bit, bool value )**

Set/clear individual bits on the layer mask. This makes getting an area in/out of only one layer easier.

- **void set_linear_damp ( float linear_damp )**

Set the rate at which objects stop moving in this area, if there are not any other forces moving it. The value is a fraction of its current speed, lost per second. Thus, a value of 1.0 should mean stopping immediately, and 0.0 means the object never stops.

In practice, as the fraction of speed lost gets smaller with each frame, a value of 1.0 does not mean the object will stop in exactly one second. Only when the physics calculations are done at 1 frame per second, it does stop in a second.

- **void set_monitorable ( bool enable )**

Set whether this area can be detected by other, monitoring, areas. Only areas need to be marked as monitorable. Bodies are always so.

- **void set_priority ( float priority )**

Set the order in which the area is processed. Greater values mean the area gets processed first. This is useful for areas which have an space override different from AREA_SPACE_OVERRIDE_DISABLED or AREA_SPACE_OVERRIDE_COMBINE, as they replace values, and are thus order-dependent.
Areas with the same priority value get evaluated in an unpredictable order, and should be differentiated if evaluation order is to be important.

- **void set_space_override_mode ( int enable )**

Set the space override mode. This mode controls how an area affects gravity and damp.

AREA_SPACE_OVERRIDE_DISABLED: This area does not affect gravity/damp. These are generally areas that exist only to detect collisions, and objects entering or exiting them.

AREA_SPACE_OVERRIDE_COMBINE: This area adds its gravity/damp values to whatever has been calculated so far. This way, many overlapping areas can combine their physics to make interesting effects.

AREA_SPACE_OVERRIDE_COMBINE_REPLACE: This area adds its gravity/damp values to whatever has been calculated so far. Then stops taking into account the rest of the areas, even the default one.

AREA_SPACE_OVERRIDE_REPLACE: This area replaces any gravity/damp, even the default one, and stops taking into account the rest of the areas.

AREA_SPACE_OVERRIDE_REPLACE_COMBINE: This area replaces any gravity/damp calculated so far, but keeps calculating the rest of the areas, down to the default one.

### 12.12 Array

**Category:** Built-In Types

#### 12.12.1 Brief Description

Generic array datatype.

#### 12.12.2 Member Functions

<table>
<thead>
<tr>
<th><strong>Array</strong></th>
<th><strong>Array ( ColorArray from )</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Array</strong></td>
<td><strong>Array ( Vector3Array from )</strong></td>
</tr>
<tr>
<td><strong>Array</strong></td>
<td><strong>Array ( Vector2Array from )</strong></td>
</tr>
<tr>
<td><strong>Array</strong></td>
<td><strong>Array ( StringArray from )</strong></td>
</tr>
<tr>
<td><strong>Array</strong></td>
<td><strong>Array ( RealArray from )</strong></td>
</tr>
<tr>
<td><strong>Array</strong></td>
<td><strong>Array ( IntArray from )</strong></td>
</tr>
<tr>
<td><strong>Array</strong></td>
<td><strong>Array ( RawArray from )</strong></td>
</tr>
<tr>
<td><strong>void</strong></td>
<td><strong>append ( var value )</strong></td>
</tr>
<tr>
<td><strong>void</strong></td>
<td><strong>back ( )</strong></td>
</tr>
<tr>
<td><strong>void</strong></td>
<td><strong>clear ( )</strong></td>
</tr>
<tr>
<td><strong>int</strong></td>
<td><strong>count ( var value )</strong></td>
</tr>
<tr>
<td><strong>bool</strong></td>
<td><strong>empty ( )</strong></td>
</tr>
<tr>
<td><strong>void</strong></td>
<td><strong>erase ( var value )</strong></td>
</tr>
<tr>
<td><strong>int</strong></td>
<td><strong>find ( var what, int from=0 )</strong></td>
</tr>
<tr>
<td><strong>int</strong></td>
<td><strong>find_last ( var value )</strong></td>
</tr>
<tr>
<td><strong>void</strong></td>
<td><strong>front ( )</strong></td>
</tr>
<tr>
<td><strong>bool</strong></td>
<td><strong>has ( var value )</strong></td>
</tr>
<tr>
<td><strong>int</strong></td>
<td><strong>hash ( )</strong></td>
</tr>
<tr>
<td><strong>void</strong></td>
<td><strong>insert ( int pos, var value )</strong></td>
</tr>
</tbody>
</table>

Continued on next page
Table 12.7 – continued from previous page

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>invert ()</td>
</tr>
<tr>
<td>bool</td>
<td>is_shared ()</td>
</tr>
<tr>
<td>void</td>
<td>pop_back ()</td>
</tr>
<tr>
<td>void</td>
<td>pop_front ()</td>
</tr>
<tr>
<td>void</td>
<td>push_back ( var value )</td>
</tr>
<tr>
<td>void</td>
<td>push_front ( var value )</td>
</tr>
<tr>
<td>void</td>
<td>remove ( int pos )</td>
</tr>
<tr>
<td>void</td>
<td>resize ( int pos )</td>
</tr>
<tr>
<td>int</td>
<td>rfind ( var what, int from=-1 )</td>
</tr>
<tr>
<td>int</td>
<td>size ()</td>
</tr>
<tr>
<td>void</td>
<td>sort ()</td>
</tr>
<tr>
<td>void</td>
<td>sort_custom ( Object obj, String func )</td>
</tr>
</tbody>
</table>

### 12.12.3 Description

Generic array, contains several elements of any type, accessible by numerical index starting at 0. Negative indices can be used to count from the right, like in Python. Arrays are always passed by reference.

### 12.12.4 Member Function Description

- **Array Array ( ColorArray from )**
  Construct an array from a RawArray.
- **Array Array ( Vector3Array from )**
  Construct an array from a RawArray.
- **Array Array ( Vector2Array from )**
  Construct an array from a RawArray.
- **Array Array ( StringArray from )**
  Construct an array from a RawArray.
- **Array Array ( RealArray from )**
  Construct an array from a RawArray.
- **Array Array ( IntArray from )**
  Construct an array from a RawArray.
- **Array Array ( RawArray from )**
  Construct an array from a RawArray.
- **void append ( var value )**
  Append an element at the end of the array (alias of push_back).
- **void back ()**
  Returns the last element of the array if the array is not empty (size>0).
- **void clear ()**
  Clear the array (resize to 0).
- **int count ( var value )**
Return the amount of times an element is in the array.

- **bool empty ()**
  Return true if the array is empty (size==0).

- **void erase ( var value )**
  Remove the first occurrence of a value from the array.

- **int find ( var what, int from=0 )**
  Searches the array for a value and returns its index or -1 if not found. Optionally, the initial search index can be passed.

- **int find_last ( var value )**
  Searches the array in reverse order for a value and returns its index or -1 if not found.

- **void front ( )**
  Returns the first element of the array if the array is not empty (size>0).

- **bool has ( var value )**
  Return true if the array contains given value.

```c++
[ "inside", 7 ].has("inside") == true
[ "inside", 7 ].has("outside") == false
[ "inside", 7 ].has(7) == true
[ "inside", 7 ].has("7") == false
```

- **int hash ( )**
  Return a hashed integer value representing the array contents.

- **void insert ( int pos, var value )**
  Insert a new element at a given position in the array. The position must be valid, or at the end of the array (pos==size()).

- **void invert ( )**
  Reverse the order of the elements in the array (so first element will now be the last).

- **bool is_shared ( )**
  Get whether this is a shared array instance.

- **void pop_back ( )**
  Remove the last element of the array.

- **void pop_front ( )**
  Remove the first element of the array.

- **void push_back ( var value )**
  Append an element at the end of the array.

- **void push_front ( var value )**
  Add an element at the beginning of the array.

- **void remove ( int pos )**
  Remove an element from the array by index.
- void `resize(int pos)`

Resize the array to contain a different number of elements. If the array size is smaller, elements are cleared, if bigger, new elements are Null.

- int `rfind(var what, int from=-1)`

Searches the array in reverse order. Optionally, a start search index can be passed. If negative, the start index is considered relative to the end of the array.

- int `size()`

Return the amount of elements in the array.

- void `sort()`

Sort the array using natural order.

- void `sort_custom(Object obj, String func)`

Sort the array using a custom method. The arguments are an object that holds the method and the name of such method. The custom method receives two arguments (a pair of elements from the array) and must return true if the first argument is less than the second, and return false otherwise.

### 12.13 AStar

Inherits: Reference < Object

Category: Core

#### 12.13.1 Brief Description

#### 12.13.2 Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>_compute_cost(int from_id, int to_id)</code></td>
<td>virtual</td>
</tr>
<tr>
<td><code>_estimate_cost(int from_id, int to_id)</code></td>
<td>virtual</td>
</tr>
<tr>
<td><code>add_point(int id, Vector3 pos, float weight_scale=1)</code></td>
<td></td>
</tr>
<tr>
<td><code>are_points_connected(int id, int to_id)</code></td>
<td>const</td>
</tr>
<tr>
<td><code>clear()</code></td>
<td></td>
</tr>
<tr>
<td><code>connect_points(int id, int to_id, bool bidirectional=true)</code></td>
<td></td>
</tr>
<tr>
<td><code>disconnect_points(int id, int to_id)</code></td>
<td></td>
</tr>
<tr>
<td><code>get_available_point_id()</code></td>
<td>const</td>
</tr>
<tr>
<td><code>get_closest_point(Vector3 to_pos)</code></td>
<td>const</td>
</tr>
<tr>
<td><code>get_closest_pos_in_segment(Vector3 to_pos)</code></td>
<td>const</td>
</tr>
<tr>
<td><code>get_id_path(int from_id, int to_id)</code></td>
<td></td>
</tr>
<tr>
<td><code>get_point_path(int from_id, int to_id)</code></td>
<td></td>
</tr>
<tr>
<td><code>get_point_pos(int id)</code></td>
<td>const</td>
</tr>
<tr>
<td><code>get_point_weight_scale(int id)</code></td>
<td>const</td>
</tr>
<tr>
<td><code>has_point(int id)</code></td>
<td>const</td>
</tr>
<tr>
<td><code>remove_point(int id)</code></td>
<td></td>
</tr>
</tbody>
</table>

#### 12.13.3 Member Function Description

- void `_compute_cost(int from_id, int to_id)` virtual
Godot Engine Documentation, Release 2.1

- void _estimate_cost (int from_id, int to_id) virtual
- void add_point (int id, Vector3 pos, float weight_scale=1)

Add a new point at the given position. The weight_scale has to be 1 or larger.

- bool are_points_connected (int id, int to_id) const
- void clear ()
- void connect_points (int id, int to_id, bool bidirectional=true)
- void disconnect_points (int id, int to_id)
- int get_available_point_id () const
- int get_closest_point (Vector3 to_pos) const
- Vector3 get_closest_pos_in_segment (Vector3 to_pos) const
- IntArray get_id_path (int from_id, int to_id)
- Vector3Array get_point_path (int from_id, int to_id)
- Vector3 get_point_pos (int id) const
- float get_point_weight_scale (int id) const
- bool has_point (int id) const
- void remove_point (int id)

12.14 AtlasTexture

Inherits: Texture < Resource < Reference < Object

Category: Core

12.14.1 Brief Description

12.14.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
<th>Modifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>Texture</td>
<td>get_atlas () const</td>
<td>const</td>
</tr>
<tr>
<td>Rect2</td>
<td>get_margin () const</td>
<td>const</td>
</tr>
<tr>
<td>Rect2</td>
<td>get_region () const</td>
<td>const</td>
</tr>
<tr>
<td>void</td>
<td>set_atlas (Texture atlas)</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>set_margin (Rect2 margin)</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>set_region (Rect2 region)</td>
<td></td>
</tr>
</tbody>
</table>

12.14.3 Signals

- atlas_changed ()
12.14.4 Member Function Description

- Texture get_atlas () const
- Rect2 get_margin () const
- Rect2 get_region () const
- void set_atlas ( Texture atlas )
- void set_margin ( Rect2 margin )
- void set_region ( Rect2 region )

12.15 AudioServer

Inherits: Object
Inherited By: AudioServerSW
Category: Core

12.15.1 Brief Description

Server interface for low level audio access.

12.15.2 Member Functions

| void | free_rid ( RID rid ) |
| float | get_event_voice_global_volume_scale () const |
| float | get_fx_global_volume_scale () const |
| float | get_stream_global_volume_scale () const |
| RID | sample_create ( int format, bool stereo, int length ) |
| RawArray | sample_get_data ( RID sample ) const |
| String | sample_get_description ( RID sample ) const |
| int | sample_get_format ( RID sample ) const |
| int | sample_get_length ( RID sample ) const |
| int | sample_get_loop_begin ( RID sample ) const |
| int | sample_get_loop_end ( RID sample ) const |
| int | sample_get_loop_format ( RID sample ) const |
| int | sample_get_mix_rate ( RID sample ) const |
| bool | sample_is_stereo ( RID sample ) const |
| void | sample_set_data ( RID sample, RawArray data ) |
| void | sample_set_description ( RID sample, String description ) |
| void | sample_set_loop_begin ( RID sample, int pos ) |
| void | sample_set_loop_end ( RID sample, int pos ) |
| void | sample_set_loop_format ( RID sample, int loop_format ) |
| void | sample_set_mix_rate ( RID sample, int mix_rate ) |
| void | sample_set_signed_data ( RID sample, RealArray data ) |
| void | set_event_voice_global_volume_scale ( float scale ) |
| void | set_fx_global_volume_scale ( float scale ) |

Continued on next page
### Table 12.8 – continued from previous page

<table>
<thead>
<tr>
<th>void</th>
<th>set_stream_global_volume_scale ( float scale )</th>
</tr>
</thead>
<tbody>
<tr>
<td>RID</td>
<td>voice_create ( )</td>
</tr>
<tr>
<td>float</td>
<td>voice_get_chorus ( RID voice ) const</td>
</tr>
<tr>
<td>float</td>
<td>voice_get_filter_cutoff ( RID voice ) const</td>
</tr>
<tr>
<td>float</td>
<td>voice_get_filter_resonance ( RID voice ) const</td>
</tr>
<tr>
<td>int</td>
<td>voice_get_filter_type ( RID voice ) const</td>
</tr>
<tr>
<td>int</td>
<td>voice_get_mix_rate ( RID voice ) const</td>
</tr>
<tr>
<td>float</td>
<td>voice_get_pan ( RID voice ) const</td>
</tr>
<tr>
<td>float</td>
<td>voice_get_pan_depth ( RID voice ) const</td>
</tr>
<tr>
<td>float</td>
<td>voice_get_pan_height ( RID voice ) const</td>
</tr>
<tr>
<td>int</td>
<td>voice_get_reverb_type ( RID voice ) const</td>
</tr>
<tr>
<td>float</td>
<td>voice_get_reverb ( RID voice ) const</td>
</tr>
<tr>
<td>float</td>
<td>voice_get_volume ( RID voice ) const</td>
</tr>
<tr>
<td>bool</td>
<td>voice_is_positional ( RID voice ) const</td>
</tr>
<tr>
<td>void</td>
<td>voice_play ( RID voice, RID sample )</td>
</tr>
<tr>
<td>void</td>
<td>voice_set_chorus ( RID voice, float chorus )</td>
</tr>
<tr>
<td>void</td>
<td>voice_set_filter ( RID voice, int type, float cutoff, float resonance, float gain=0 )</td>
</tr>
<tr>
<td>void</td>
<td>voice_set_mix_rate ( RID voice, int rate )</td>
</tr>
<tr>
<td>void</td>
<td>voice_set_pan ( RID voice, float pan, float depth=0, float height=0 )</td>
</tr>
<tr>
<td>void</td>
<td>voice_set_positional ( RID voice, bool enabled )</td>
</tr>
<tr>
<td>void</td>
<td>voice_set_reverb ( RID voice, int room, float reverb )</td>
</tr>
<tr>
<td>void</td>
<td>voice_set_volume ( RID voice, float volume )</td>
</tr>
<tr>
<td>void</td>
<td>voice_stop ( RID voice )</td>
</tr>
</tbody>
</table>

#### 12.15.3 Numeric Constants

- **SAMPLE_FORMAT_PCM8 = 0** — Sample format is 8 bits, signed.
- **SAMPLE_FORMAT_PCM16 = 1** — Sample format is 16 bits, little-endian, signed.
- **SAMPLE_FORMAT.IMA_ADPCM = 2** — Sample format is IMA-ADPCM compressed.
- **SAMPLE_LOOP_NONE = 0** — Sample does not loop.
- **SAMPLE_LOOP_FORWARD = 1** — Sample loops in forward mode.
- **SAMPLE_LOOP_PING_PONG = 2** — Sample loops in a bidirectional way.
- **FILTER_NONE = 0** — Filter is disabled.
- **FILTER_LOWPASS = 1** — Filter is a resonant lowpass.
- **FILTER_BANDPASS = 2** — Filter is a resonant bandpass.
- **FILTER_HIPASS = 3** — Filter is a resonant highpass.
- **FILTER_NOTCH = 4** — Filter is a notch (band reject).
- **FILTER_BANDLIMIT = 6** — Filter is a bandlimit (resonance used as highpass).
- **REVERB_SMALL = 0** — Small reverb room (closet, bathroom, etc).
- **REVERB_MEDIUM = 1** — Medium reverb room (living room).
- **REVERB_LARGE = 2** — Large reverb room (warehouse).
- **REVERB_HALL = 3** — Large reverb room with long decay.
12.15.4 Description

AudioServer is a low level server interface for audio access. It is in charge of creating sample data (playable audio) as well as its playback via a voice interface.

12.15.5 Member Function Description

- void `free_rid ( RID rid )`
  Free a `RID` resource.
- `float get_event_voice_global_volume_scale ( ) const`
  Return the global scale for event-based stream playback.
- `float get_fx_global_volume_scale ( ) const`
  Return the global scale for all voices.
- `float get_stream_global_volume_scale ( ) const`
  Return the global scale for stream playback.
- `RID sample_create ( int format, bool stereo, int length )`
  Create an audio sample, return a `RID` referencing it. The sample will be created with a given format (from the SAMPLE_FORMAT_* enum), a total length (in samples, not bytes), in either stereo or mono.
  Even if a stereo sample consists of a left sample and a right sample, it still counts as one sample for length purposes.
- `RawArray sample_get_data ( RID sample ) const`
  Return the sample data as an array of bytes. The length will be the expected length in bytes.
- `String sample_get_description ( RID sample ) const`
  Return the description of an audio sample. Mainly used for organization.
- `int sample_get_format ( RID sample ) const`
  Return the format of the audio sample, in the form of the SAMPLE_FORMAT_* enum.
- `int sample_get_length ( RID sample ) const`
  Return the length in samples (not bytes) of the audio sample. Even if a stereo sample consists of a left sample and a right sample, it still counts as one sample for length purposes.
- `int sample_get_loop_begin ( RID sample ) const`
  Return the initial loop point of a sample. Only has effect if sample loop is enabled. See `sample_set_loop_format`.
- `int sample_get_loop_end ( RID sample ) const`
  Return the final loop point of a sample. Only has effect if sample loop is enabled. See `sample_set_loop_format`.
- `int sample_get_loop_format ( RID sample ) const`
  Return the loop format for a sample, as a value from the SAMPLE_LOOP_* enum.
- `int sample_get_mix_rate ( RID sample ) const`
  Return the mix rate of the given sample.
- `bool sample_is_stereo ( RID sample ) const`
  Return whether the sample is stereo (2 channels).
Set the sample data for a given sample as an array of bytes. The length must be equal to the sample length expected in bytes or an error will be produced. The byte length can be calculated as follows:

- If the sample format is SAMPLE_FORMAT_PCM16, multiply it by 2.
- If the sample format is SAMPLE_FORMAT_IMA_ADPCM, divide it by 2 (rounding any fraction up), then add 4.
- If the sample is stereo (sample_is_stereo), multiply it by 2.

Set the description of an audio sample. Mainly used for organization.

Set the initial loop point of a sample. Only has effect if sample loop is enabled. See sample_set_loop_format.

Set the final loop point of a sample. Only has effect if sample loop is enabled. See sample_set_loop_format.

Set the loop format for a sample from the SAMPLE_LOOP_* enum. As a warning, Ping Pong loops may not be available on some hardware-mixing platforms.

Change the default mix rate of a given sample.

Set the sample data for a given sample as an array of floats. The length must be equal to the sample length or an error will be produced.

For this method, a stereo sample is made from two samples. Thus, in case of a stereo sample, the array length must be twice the length returned by sample_get_length.

Trying to alter a SAMPLE_FORMAT_IMA_ADPCM sample is not supported. It will throw an error to the console, but will not alter the sample data.

Set global scale for event-based stream (EventStream) playback. Default is 1.0.

Set global scale for all voices (not including streams). Default is 1.0.

Set global scale for stream playback. Default is 1.0.

Allocate a voice for playback. Voices are persistent. A voice can play a single sample at the same time. See sample_create.

Return the current chorus send for a given voice (0 to 1).

Return the current filter cutoff (in hz) for a given voice.
• float voice_get_filter_resonance ( RID voice ) const
Return the current filter resonance for a given voice.

• int voice_get_filter_type ( RID voice ) const
Return the current selected filter type for a given voice, from the FILTER_* enum.

• int voice_get_mix_rate ( RID voice ) const
Return the current mix rate for a given voice.

• float voice_get_pan ( RID voice ) const
Return the current pan for a given voice (-1 to +1 range).

• float voice_get_pan_depth ( RID voice ) const
Return the current pan depth for a given voice (-1 to +1 range).

• float voice_get_pan_height ( RID voice ) const
Return the current pan height for a given voice (-1 to +1 range).

• float voice_get_reverb ( RID voice ) const
Return the current reverb send for a given voice (0 to 1).

• int voice_get_reverb_type ( RID voice ) const
Return the current reverb type for a given voice from the REVERB_* enum.

• float voice_get_volume ( RID voice ) const
Return the current volume for a given voice.

• bool voice_is_positional ( RID voice ) const
Return whether the current voice is positional. See voice_set_positional.

• void voice_play ( RID voice, RID sample )
Start playback of a given voice using a given sample. If the voice was already playing it will be restarted.

• void voice_set_chorus ( RID voice, float chorus )
Set chorus send post processing for the voice (from 0 to 1).

• void voice_set_filter ( RID voice, int type, float cutoff, float resonance, float gain=0 )
Set a resonant filter post processing for the voice. Filter type is a value from the FILTER_* enum.

• void voice_set_mix_rate ( RID voice, int rate )
Set a different playback mix rate for the given voice.

• void voice_set_pan ( RID voice, float pan, float depth=0, float height=0 )
Change the pan of a currently playing voice and, optionally, the depth and height for a positional/3D sound. Panning values are expressed within the -1 to +1 range.

• void voice_set_positional ( RID voice, bool enabled )
Set whether a given voice is positional. This is only interpreted as a hint and used for backends that may support binaural encoding.

• void voice_set_reverb ( RID voice, int room, float reverb )
Set the reverb send post processing for the voice (from 0 to 1) and the reverb type, from the REVERB_* enum.
Change the volume of a currently playing voice. Volume is expressed as linear gain where 0.0 is mute and 1.0 is default.

- void voice_stop ( RID voice )

Stop a given voice.

12.16 AudioServerSW

Inherits: AudioServer < Object
Category: Core

12.16.1 Brief Description
Software implementation of AudioServer.

12.16.2 Description
This is a software audio server. It does not use any kind of hardware acceleration.
This class does not expose any new method.

12.17 AudioStream

Inherits: Resource < Reference < Object
Inherited By: AudioStreamSpeex, AudioStreamMPC, AudioStreamOGGVorbis, AudioStreamOpus
Category: Core

12.17.1 Brief Description
Base class for audio streams.

12.17.2 Description
Base class for audio streams. Audio streams are used for music playback, or other types of streamed sounds that don’t fit or require more flexibility than a Sample.

12.18 AudioStreamMPC

Inherits: AudioStream < Resource < Reference < Object
Category: Core
12.18.1 Brief Description
MusePack audio stream driver.

12.18.2 Description
MusePack audio stream driver.

12.19 AudioStreamOGGVorbis

Inherits: AudioStream < Resource < Reference < Object
Category: Core

12.19.1 Brief Description
OGG Vorbis audio stream driver.

12.19.2 Description
OGG Vorbis audio stream driver.

12.20 AudioStreamOpus

Inherits: AudioStream < Resource < Reference < Object
Category: Core

12.20.1 Brief Description
Opus Codec audio stream driver.

12.20.2 Description
Opus Codec audio stream driver.

12.21 AudioStreamPlayback

Inherits: Reference < Object
Category: Core
12.21.1 Brief Description

12.21.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td><code>get_channels ( ) const</code></td>
</tr>
<tr>
<td>float</td>
<td><code>get_length ( ) const</code></td>
</tr>
<tr>
<td>int</td>
<td><code>get_loop_count ( ) const</code></td>
</tr>
<tr>
<td>int</td>
<td><code>get_minimum_buffer_size ( ) const</code></td>
</tr>
<tr>
<td>int</td>
<td><code>get_mix_rate ( ) const</code></td>
</tr>
<tr>
<td>float</td>
<td><code>get_pos ( ) const</code></td>
</tr>
<tr>
<td>bool</td>
<td><code>has_loop ( ) const</code></td>
</tr>
<tr>
<td>bool</td>
<td><code>is_playing ( ) const</code></td>
</tr>
<tr>
<td>void</td>
<td><code>play (float from_pos_sec=0)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>seek_pos (float pos)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_loop (bool enabled)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>stop ( )</code></td>
</tr>
</tbody>
</table>

12.21.3 Member Function Description

- `int get_channels ( ) const`
- `float get_length ( ) const`
- `int get_loop_count ( ) const`
- `int get_minimum_buffer_size ( ) const`
- `int get_mix_rate ( ) const`
- `float get_pos ( ) const`
- `bool has_loop ( ) const`
- `bool is_playing ( ) const`
- `void play (float from_pos_sec=0)`
- `void seek_pos (float pos)`
- `void set_loop (bool enabled)`
- `void stop ( )`

12.22 AudioStreamSpeex

Inherits: `AudioStream < Resource < Reference < Object`

Category: Core

12.22.1 Brief Description

Speex audio stream driver.
12.22.2 Description

Speex audio stream driver. Speex is very useful for compressed speech. It allows loading a very large amount of speech in memory at little IO/latency cost.

12.23 BackBufferCopy

Inherits: Node2D < CanvasItem < Node < Object

Category: Core

12.23.1 Brief Description

Copies a region of the screen (or the whole screen) to a buffer so it can be accessed with the texscreen() shader instruction.

12.23.2 Member Functions

| int | get_copy_mode ( ) const |
| Rect2 | get_rect ( ) const |
| void | set_copy_mode ( int copy_mode ) |
| void | set_rect ( Rect2 rect ) |

12.23.3 Numeric Constants

- **COPY_MODE_DISABLED** = 0 — Disables the buffering mode. This means the BackBufferCopy node will directly use the portion of screen it covers.
- **COPY_MODE_RECT** = 1 — Sets the copy mode to a region.
- **COPY_MODE_VIEWPORT** = 2 — Sets the copy mode to the entire screen.

12.23.4 Description

Node for back-buffering the currently displayed screen. The region defined in the BackBufferCopy node is bufferized with the content of the screen it covers, or the entire screen according to the copy mode set. Accessing this buffer is done with the texscreen() shader instruction.

12.23.5 Member Function Description

- **int get_copy_mode ( ) const**
  Return the copy mode currently applied to the BackBufferCopy (refer to constants section).
- **Rect2 get_rect ( ) const**
  Return the area covered by the BackBufferCopy.
- **void set_copy_mode ( int copy_mode )**
  Set the copy mode of the BackBufferCopy (refer to constants section).
- void set_rect ( Rect2 rect )

Defines the area covered by the BackBufferCopy.

## 12.24 BakedLight

Inherits: Resource < Reference < Object

Category: Core

### 12.24.1 Brief Description

### 12.24.2 Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void add_lightmap ( Texture texture, Vector2 gen_size )</td>
<td></td>
</tr>
<tr>
<td>void clear_lightmaps ()</td>
<td></td>
</tr>
<tr>
<td>void erase_lightmap ( int id )</td>
<td></td>
</tr>
<tr>
<td>float get_ao_radius () const</td>
<td></td>
</tr>
<tr>
<td>float get_ao_strength () const</td>
<td></td>
</tr>
<tr>
<td>bool get_bake_flag ( int flag ) const</td>
<td></td>
</tr>
<tr>
<td>int get_bounces () const</td>
<td></td>
</tr>
<tr>
<td>float get_cell_extra_margin () const</td>
<td></td>
</tr>
<tr>
<td>int get_cell_subdivision () const</td>
<td></td>
</tr>
<tr>
<td>float get_edge_damp () const</td>
<td></td>
</tr>
<tr>
<td>float get_energy_multiplier () const</td>
<td></td>
</tr>
<tr>
<td>int get_format () const</td>
<td></td>
</tr>
<tr>
<td>int get_initial_lattice_subdiv () const</td>
<td></td>
</tr>
<tr>
<td>RawArray get_light () const</td>
<td></td>
</tr>
<tr>
<td>int get_mode () const</td>
<td></td>
</tr>
<tr>
<td>float get_normal_damp () const</td>
<td></td>
</tr>
<tr>
<td>RawArray get_octree () const</td>
<td></td>
</tr>
<tr>
<td>float get_plot_size () const</td>
<td></td>
</tr>
<tr>
<td>Color get_realtime_color () const</td>
<td></td>
</tr>
<tr>
<td>bool get_realtime_color_enabled () const</td>
<td></td>
</tr>
<tr>
<td>float get_realtime_energy () const</td>
<td></td>
</tr>
<tr>
<td>IntArray get_sampler_octree () const</td>
<td></td>
</tr>
<tr>
<td>float get_saturation () const</td>
<td></td>
</tr>
<tr>
<td>float get_tint () const</td>
<td></td>
</tr>
<tr>
<td>bool get_transfer_lightmaps_only_to_uv2 () const</td>
<td></td>
</tr>
<tr>
<td>void set_ao_radius ( float ao_radius )</td>
<td></td>
</tr>
<tr>
<td>void set_ao_strength ( float ao_strength )</td>
<td></td>
</tr>
<tr>
<td>void set_bake_flag ( int flag, bool enabled )</td>
<td></td>
</tr>
<tr>
<td>void set_bounces ( int bounces )</td>
<td></td>
</tr>
<tr>
<td>void set_cell_extra_margin ( float cell_extra_margin )</td>
<td></td>
</tr>
<tr>
<td>void set_cell_subdivision ( int cell_subdivision )</td>
<td></td>
</tr>
<tr>
<td>void set_edge_damp ( float edge_damp )</td>
<td></td>
</tr>
<tr>
<td>void set_energy_multiplier ( float energy_multiplier )</td>
<td></td>
</tr>
<tr>
<td>void set_format ( int format )</td>
<td></td>
</tr>
</tbody>
</table>

Continued on next page
Table 12.9 – continued from previous page

<table>
<thead>
<tr>
<th>Function Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void set_gamma_adjust (float gamma_adjust)</td>
</tr>
<tr>
<td>void set_initial_lattice_subdiv (int cell_subdivision)</td>
</tr>
<tr>
<td>void set_light (RawArray light)</td>
</tr>
<tr>
<td>void set_mode (int mode)</td>
</tr>
<tr>
<td>void set_normal_damp (float normal_damp)</td>
</tr>
<tr>
<td>void set_octree (RawArray octree)</td>
</tr>
<tr>
<td>void set_plot_size (float plot_size)</td>
</tr>
<tr>
<td>void set_realtime_color (Color tint)</td>
</tr>
<tr>
<td>void set_realtime_color_enabled (bool enabled)</td>
</tr>
<tr>
<td>void set_realtime_energy (float energy)</td>
</tr>
<tr>
<td>void set_sampler_octree (IntArray sampler_octree)</td>
</tr>
<tr>
<td>void set_saturation (float saturation)</td>
</tr>
<tr>
<td>void set_tint (float tint)</td>
</tr>
<tr>
<td>void set_transfer_lightmaps_only_to_uv2 (bool enable)</td>
</tr>
</tbody>
</table>

12.24.3 Numeric Constants

- MODE_OCTREE = 0
- MODE_LIGHTMAPS = 1
- BAKE_DIFFUSE = 0
- BAKE_SPECULAR = 1
- BAKE_TRANSLUCENT = 2
- BAKE_CONSERVE_ENERGY = 3
- BAKE_MAX = 5

12.24.4 Member Function Description

- void add_lightmap (Texture texture, Vector2 gen_size)
- void clear_lightmaps ()
- void erase_lightmap (int id)
- float get_ao_radius () const
- float get_ao_strength () const
- bool get_bake_flag (int flag) const
- int get_bounces () const
- float get_cell_extra_margin () const
- int get_cell_subdivision () const
- float get_edge_damp () const
- float get_energy_multiplier () const
- int get_format () const
- float get_gamma_adjust () const
- int get_initial_lattice_subdiv () const
- `RawArray` `get_light()` `const`
- `int` `get_mode()` `const`
- `float` `get_normal_damp()` `const`
- `RawArray` `get_octree()` `const`
- `float` `get_plot_size()` `const`
- `Color` `get_realtime_color()` `const`
- `bool` `get_realtime_color_enabled()` `const`
- `float` `get_realtime_energy()` `const`
- `IntArray` `get_sampler_octree()` `const`
- `float` `get_saturation()` `const`
- `float` `get_tint()` `const`
- `bool` `get_transfer_lightmaps_only_to_uv2()` `const`
- `void` `set_ao_radius()` `float` `ao_radius`
- `void` `set_ao_strength()` `float` `ao_strength`
- `void` `set_bake_flag()` `int` `flag`, `bool` `enabled`
- `void` `set_bounces()` `int` `bounces`
- `void` `set_cell_extra_margin()` `float` `cell_extra_margin`
- `void` `set_cell_subdivision()` `int` `cell_subdivision`
- `void` `set_edge_damp()` `float` `edge_damp`
- `void` `set_energy_multiplier()` `float` `energy_multiplier`
- `void` `set_format()` `int` `format`
- `void` `set_gamma_adjust()` `float` `gamma_adjust`
- `void` `set_initial_lattice_subdiv()` `int` `cell_subdivision`
- `void` `set_light()` `RawArray` `light`
- `void` `set_mode()` `int` `mode`
- `void` `set_normal_damp()` `float` `normal_damp`
- `void` `set_octree()` `RawArray` `octree`
- `void` `set_plot_size()` `float` `plot_size`
- `void` `set_realtime_color()` `Color` `tint`
- `void` `set_realtime_color_enabled()` `bool` `enabled`
- `void` `set_realtime_energy()` `float` `energy`
- `void` `set_sampler_octree()` `IntArray` `sampler_octree`
- `void` `set_saturation()` `float` `saturation`
- `void` `set_tint()` `float` `tint`
- `void` `set_transfer_lightmaps_only_to_uv2()` `bool` `enable`
12.25 BakedLightInstance

Inherits: VisualInstance < Spatial < Node < Object
Category: Core

12.25.1 Brief Description

12.25.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>get_baked_light ( ) const</td>
</tr>
<tr>
<td>RID</td>
<td>get_baked_light_instance ( ) const</td>
</tr>
<tr>
<td>void</td>
<td>set_baked_light ( Object baked_light )</td>
</tr>
</tbody>
</table>

12.25.3 Signals

- baked_light_changed ( )

12.25.4 Member Function Description

- Object get_baked_light ( ) const
- RID get_baked_light_instance ( ) const
- void set_baked_light ( Object baked_light )

12.26 BakedLightSampler

Inherits: VisualInstance < Spatial < Node < Object
Category: Core

12.26.1 Brief Description

12.26.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>float</td>
<td>get_param ( int param ) const</td>
</tr>
<tr>
<td>int</td>
<td>get_resolution ( ) const</td>
</tr>
<tr>
<td>void</td>
<td>set_param ( int param, float value )</td>
</tr>
<tr>
<td>void</td>
<td>set_resolution ( int resolution )</td>
</tr>
</tbody>
</table>

12.26.3 Numeric Constants

- PARAM_RADIUS = 0
- PARAM_STRENGTH = 1
- PARAM_ATTENUATION = 2
- PARAM_DETAIL_RATIO = 3
• PARAM_MAX = 4

12.26.4 Member Function Description

- float get_param ( int param ) const
- int get_resolution ( ) const
- void set_param ( int param, float value )
- void set_resolution ( int resolution )

12.27 BaseButton

Inherits: Control < CanvasItem < Node < Object

Inherited By: LinkButton, TextureButton, Button

Category: Core

12.27.1 Brief Description

Provides a base class for different kinds of buttons.

12.27.2 Member Functions

<table>
<thead>
<tr>
<th>void _pressed ( ) virtual</th>
</tr>
</thead>
<tbody>
<tr>
<td>void _toggled ( bool pressed ) virtual</td>
</tr>
<tr>
<td>bool get_click_on_press ( ) const</td>
</tr>
<tr>
<td>int get_draw_mode ( ) const</td>
</tr>
<tr>
<td>int get_enabled_focus_mode ( ) const</td>
</tr>
<tr>
<td>Object get_shortcut ( ) const</td>
</tr>
<tr>
<td>bool is_disabled ( ) const</td>
</tr>
<tr>
<td>bool is_hovered ( ) const</td>
</tr>
<tr>
<td>bool is_pressed ( ) const</td>
</tr>
<tr>
<td>bool is_toggle_mode ( ) const</td>
</tr>
<tr>
<td>void set_click_on_press ( bool enable )</td>
</tr>
<tr>
<td>void set_disabled ( bool disabled )</td>
</tr>
<tr>
<td>void set_enabled_focus_mode ( int mode )</td>
</tr>
<tr>
<td>void set_pressed ( bool pressed )</td>
</tr>
<tr>
<td>void set_shortcut ( Object shortcut )</td>
</tr>
<tr>
<td>void set_toggle_mode ( bool enabled )</td>
</tr>
</tbody>
</table>

12.27.3 Signals

- button_down ( )
  Emitted when the button starts being held down.
- button_up ( )
  Emitted when the button stops being held down.
• pressed ( )
  This signal is emitted every time the button is toggled or pressed (i.e. activated, so on button_down if “Click on press” is active and on button_up otherwise).

• released ( )
  Emitted when the button was released. This is only emitted by non-toggle buttons and if “Click on press” is active.

• toggled ( bool pressed )
  This signal is emitted when the button was just toggled between pressed and normal states (only if toggle_mode is active). The new state is contained in the pressed argument.

### 12.27.4 Numeric Constants

- DRAW_NORMAL = 0 — The normal state (i.e. not pressed, not hovered, not toggled and enabled) of buttons.
- DRAW_PRESSED = 1 — The state of buttons are pressed.
- DRAW_HOVER = 2 — The state of buttons are hovered.
- DRAW.Disabled = 3 — The state of buttons are disabled.

### 12.27.5 Description

BaseButton is the abstract base class for buttons, so it shouldn’t be used directly (It doesn’t display anything). Other types of buttons inherit from it.

### 12.27.6 Member Function Description

- void _pressed ( ) virtual
  Called when button is pressed.

- void _toggled ( bool pressed ) virtual
  Called when button is toggled (only if toggle_mode is active).

- bool get_click_on_press ( ) const
  Return the state of the click_on_press property (see set_click_on_press).

- int get_draw_mode ( ) const
  Return the visual state used to draw the button. This is useful mainly when implementing your own draw code by either overriding _draw() or connecting to “draw” signal. The visual state of the button is defined by the DRAW_* enum.

- int get_enabled_focus_mode ( ) const
  Returns focus access mode used when switching between enabled/disabled (see Control.set_focus_mode and set_disabled).

- Object get_shortcut ( ) const

- bool is_disabled ( ) const
  Return whether the button is in disabled state (see set_disabled).

- bool is_hovered ( ) const
Return true if mouse entered the button before it exit.

- `bool is_pressed ( ) const`

If toggle_mode is active, return whether the button is toggled. If toggle_mode is not active, return whether the button is pressed down.

- `bool is_toggle_mode ( ) const`

Return the toggle_mode property (see `set_toggle_mode`).

- `void set_click_on_press ( bool enable )`

Set the button click_on_press mode. This mode generates click events when a mouse button or key is just pressed (by default events are generated when the button/keys are released and both press and release occur in the visual area of the Button).

- `void set_disabled ( bool disabled )`

Set the button into disabled state. When a button is disabled, it can’t be clicked or toggled.

- `void set_enabled_focus_mode ( int mode )`

Sets the focus access mode to use when switching between enabled/disabled (see `Control.set_focus_mode` and `set_disabled`).

- `void set_pressed ( bool pressed )`

Set the button to pressed state (only if toggle_mode is active).

- `void set_shortcut ( Object shortcut )`

- `void set_toggle_mode ( bool enabled )`

Set the button toggle_mode property. Toggle mode makes the button flip state between pressed and unpressed each time its area is clicked.

## 12.28 BitMap

**Inherits:** Resource < Reference < Object

**Category:** Core

### 12.28.1 Brief Description

### 12.28.2 Member Functions

<table>
<thead>
<tr>
<th>void</th>
<th>create ( Vector2 size )</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>create_from_image_alpha ( Image image )</td>
</tr>
<tr>
<td>bool</td>
<td>get_bit ( Vector2 pos ) const</td>
</tr>
<tr>
<td>Vector2</td>
<td>get_size ( ) const</td>
</tr>
<tr>
<td>int</td>
<td>get_true_bit_count ( ) const</td>
</tr>
<tr>
<td>void</td>
<td>set_bit ( Vector2 pos, bool bit )</td>
</tr>
<tr>
<td>void</td>
<td>set_bit_rect ( Rect2 p_rect, bool bit )</td>
</tr>
</tbody>
</table>
12.28.3 Member Function Description

• void create ( Vector2 size )
• void create_from_image_alpha ( Image image )
• bool get_bit ( Vector2 pos ) const
• Vector2 get_size ( ) const
• int get_true_bit_count ( ) const
• void set_bit ( Vector2 pos, bool bit )
• void set_bit_rect ( Rect2 p_rect, bool bit )

12.29 BitmapFont

Inherits: Font < Resource < Reference < Object
Category: Core

12.29.1 Brief Description

12.29.2 Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void add_char ( int character, int texture, Rect2 rect, Vector2 align=Vector2(0, 0), float advance=-1 )</td>
<td>Add a character to the font, where character is the unicode value, texture is the texture index, rect is the region in the texture (in pixels!), align is the (optional) alignment for the character and advance is the (optional) advance.</td>
</tr>
<tr>
<td>void add_kerning_pair ( int char_a, int char_b, int kerning )</td>
<td>Add a kerning pair to the BitmapFont as a difference. Kerning pairs are special cases where a typeface advance is determined by the next character.</td>
</tr>
<tr>
<td>void add_texture ( Texture texture )</td>
<td>Add a texture to the BitmapFont.</td>
</tr>
<tr>
<td>void clear ( )</td>
<td></td>
</tr>
<tr>
<td>int create_from_fnt ( String path )</td>
<td></td>
</tr>
<tr>
<td>Vector2 get_char_size ( int char, int next=0 ) const</td>
<td></td>
</tr>
<tr>
<td>Object get_fallback ( ) const</td>
<td></td>
</tr>
<tr>
<td>int get_kerning_pair ( int char_a, int char_b ) const</td>
<td></td>
</tr>
<tr>
<td>Texture get_texture ( int idx ) const</td>
<td></td>
</tr>
<tr>
<td>int get_texture_count ( ) const</td>
<td></td>
</tr>
<tr>
<td>void set_ascent ( float px )</td>
<td></td>
</tr>
<tr>
<td>void set_distance_field_hint ( bool enable )</td>
<td></td>
</tr>
<tr>
<td>void set_fallback ( Object fallback )</td>
<td></td>
</tr>
<tr>
<td>void set_height ( float px )</td>
<td></td>
</tr>
</tbody>
</table>

12.29.3 Member Function Description

• **void add_char ( int character, int texture, Rect2 rect, Vector2 align=Vector2(0, 0), float advance=-1 )**

Add a character to the font, where `character` is the unicode value, `texture` is the texture index, `rect` is the region in the texture (in pixels!), `align` is the (optional) alignment for the character and `advance` is the (optional) advance.

• **void add_kerning_pair ( int char_a, int char_b, int kerning )**

Add a kerning pair to the **BitmapFont** as a difference. Kerning pairs are special cases where a typeface advance is determined by the next character.

• **void add_texture ( Texture texture )**

Add a texture to the **BitmapFont**.
- void clear()

Clear all the font data.

- int create_from_fnt(String path)

Return the size of a character, optionally taking kerning into account if the next character is provided.

- Object get_fallback()

- int get_kerning_pair(int char_a, int char_b)

Return a kerning pair as a difference.

- Texture get_texture(int idx)

- int get_texture_count()

- void set_ascent(float px)

Set the font ascent (number of pixels above the baseline).

- void set_distance_field_hint(bool enable)

- void set_fallback(Object fallback)

- void set_height(float px)

Set the total font height (ascent plus descent) in pixels.

## 12.30 BoneAttachment

Inherits: Spatial < Node < Object

Category: Core

### 12.30.1 Brief Description

A node that will attach to a bone.

### 12.30.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>get_bone_name() const</td>
</tr>
<tr>
<td>void</td>
<td>set_bone_name(String bone_name)</td>
</tr>
</tbody>
</table>

### 12.30.3 Description

This node must be the child of a Skeleton node. You can then select a bone for this node to attach to. The BoneAttachment node will copy the transform of the selected bone.

### 12.30.4 Member Function Description

- String get_bone_name() const

- void set_bone_name(String bone_name)
12.31 bool

Category: Built-In Types

12.31.1 Brief Description

Boolean built-in type

12.31.2 Member Functions

<table>
<thead>
<tr>
<th>bool</th>
<th>bool ( int from )</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool</td>
<td>bool ( float from )</td>
</tr>
<tr>
<td>bool</td>
<td>bool ( String from )</td>
</tr>
</tbody>
</table>

12.31.3 Description

Boolean built-in type.

12.31.4 Member Function Description

- bool bool ( int from )

  Cast an int value to a boolean value, this method will return true if called with an integer value different to 0 and false in other case.

- bool bool ( float from )

  Cast a float value to a boolean value, this method will return true if called with a floating point value different to 0 and false in other case.

- bool bool ( String from )

  Cast a String value to a boolean value, this method will return true if called with a non empty string and false in other case. Examples: bool ('False') returns true, bool (''). returns false

12.32 BoxContainer

Inherits: Container < Control < CanvasItem < Node < Object

Inherited By: VBoxContainer, ButtonGroup, HBoxContainer, ColorPicker

Category: Core

12.32.1 Brief Description

Base class for Box containers.
12.32.2 Member Functions

<table>
<thead>
<tr>
<th>void</th>
<th>add_spacer ( bool begin )</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>get_alignment ( ) const</td>
</tr>
<tr>
<td>void</td>
<td>set_alignment ( int alignment )</td>
</tr>
</tbody>
</table>

12.32.3 Numeric Constants

- ALIGN_BEGIN = 0 — Align children with beginning of the container.
- ALIGN_CENTER = 1 — Align children with center of the container.
- ALIGN_END = 2 — Align children with end of the container.

12.32.4 Description

Base class for Box containers. It arranges children controls vertically or horizontally, and rearranges them automatically when their minimum size changes.

12.32.5 Member Function Description

- void add_spacer ( bool begin )
  Add a control to the box as a spacer.
  If begin is true the spacer control will be inserted in front of other children.

- int get_alignment ( ) const
  Return the alignment of children in the container.

- void set_alignment ( int alignment )
  Set the alignment of children in the container(Must be one of ALIGN_BEGIN, ALIGN_CENTER or ALIGN_END).

12.33 BoxShape

Inherits: Shape < Resource < Reference < Object

Category: Core

12.33.1 Brief Description

Box shape resource.

12.33.2 Member Functions

<table>
<thead>
<tr>
<th>Vector3</th>
<th>get_extents ( ) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>set_extents ( Vector3 extents )</td>
</tr>
</tbody>
</table>
12.33.3 Description

Box shape resource, which can be set into a PhysicsBody or area.

12.33.4 Member Function Description

- **Vector3 get_extents ( ) const**
  Return the half extents of the shape.
- **void set_extents ( Vector3 extents )**
  Set the half extents for the shape.

12.34 Button

**Inherits:** BaseButton < Control < CanvasItem < Node < Object

**Inherited By:** OptionButton, ColorPickerButton, CheckButton, MenuButton, ToolButton, CheckBox

**Category:** Core

12.34.1 Brief Description

Standard themed Button.

12.34.2 Member Functions

<table>
<thead>
<tr>
<th>Texture</th>
<th>get_button_icon ( ) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool</td>
<td>get_clip_text ( ) const</td>
</tr>
<tr>
<td>String</td>
<td>get_text ( ) const</td>
</tr>
<tr>
<td>int</td>
<td>get_text_align ( ) const</td>
</tr>
<tr>
<td>bool</td>
<td>is_flat ( ) const</td>
</tr>
<tr>
<td>void</td>
<td>set_button_icon ( Texture texture )</td>
</tr>
<tr>
<td>void</td>
<td>set_clip_text ( bool enabled )</td>
</tr>
<tr>
<td>void</td>
<td>set_flat ( bool enabled )</td>
</tr>
<tr>
<td>void</td>
<td>set_text ( String text )</td>
</tr>
<tr>
<td>void</td>
<td>set_text_align ( int align )</td>
</tr>
</tbody>
</table>

12.34.3 Numeric Constants

- **ALIGN_LEFT = 0** — Align the text to the left.
- **ALIGN_CENTER = 1** — Center the text.
- **ALIGN_RIGHT = 2** — Align the text to the right.

12.34.4 Description

Button is the standard themed button. It can contain text and an icon, and will display them according to the current Theme.
12.34.5 Member Function Description

- **Texture** get_button_icon ( ) const
  Return the button icon.

- **bool** get_clip_text ( ) const
  Return the state of the clip_text property (see set_clip_text).

- **String** get_text ( ) const
  Return the button text.

- **int** get_text_align ( ) const
  Return the text alignment policy.

- **bool** is_flat ( ) const
  Return the state of the flat property (see set_flat).

- **void** set_button_icon ( Texture texture )
  Set the icon that will be displayed next to the text inside the button area.

- **void** set_clip_text ( bool enabled )
  Set the clip_text property of a Button. When this property is enabled, text that is too large to fit the button is clipped, when disabled (default) the Button will always be wide enough to hold the text.

- **void** set_flat ( bool enabled )
  Set the flat property of a Button. Flat buttons don’t display decoration unless hovered or pressed.

- **void** set_text ( String text )
  Set the button text, which will be displayed inside the button area.

- **void** set_text_align ( int align )
  Set the text alignment policy, using one of the ALIGN_* constants.

12.35 ButtonArray

**Inherits:** Control < CanvasItem < Node < Object

**Inherited By:** HButtonArray, VButtonArray

**Category:** Core

12.35.1 Brief Description

Array of Buttons.
12.35.2 Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void add_button (String text, String tooltip=&quot;&quot;)</td>
<td>Append a new button to the array, with the specified text.</td>
</tr>
<tr>
<td>void add_icon_button (Texture icon, String text=&quot;&quot;, String tooltip=&quot;&quot;)</td>
<td>Append a new button to the array, with the specified icon and text.</td>
</tr>
<tr>
<td>void clear ( )</td>
<td></td>
</tr>
<tr>
<td>int get_button_count ( ) const</td>
<td></td>
</tr>
<tr>
<td>Texture get_button_icon (int button_idx) const</td>
<td></td>
</tr>
<tr>
<td>String get_button_text (int button_idx) const</td>
<td></td>
</tr>
<tr>
<td>String get_button_tooltip (int button_idx) const</td>
<td></td>
</tr>
<tr>
<td>int get_hovered ( ) const</td>
<td></td>
</tr>
<tr>
<td>int get_selected ( ) const</td>
<td></td>
</tr>
<tr>
<td>bool is_flat ( ) const</td>
<td></td>
</tr>
<tr>
<td>void set_button_icon (int button_idx, Texture icon)</td>
<td></td>
</tr>
<tr>
<td>void set_button_text (int button_idx, String text)</td>
<td></td>
</tr>
<tr>
<td>void set_button_tooltip (int button_idx, String text)</td>
<td></td>
</tr>
<tr>
<td>void set_flat (bool enabled)</td>
<td></td>
</tr>
<tr>
<td>void set_selected (int button_idx)</td>
<td></td>
</tr>
</tbody>
</table>

A button has been selected, its index is given as the argument.

12.35.4 Numeric Constants

• ALIGN_BEGIN = 0 — Align buttons at the beginning.
• ALIGN_CENTER = 1 — Align buttons in the middle.
• ALIGN_END = 2 — Align buttons at the end.
• ALIGN_FILL = 3 — Spread the buttons, but keep them small.
• ALIGN_EXPAND_FILL = 4 — Spread the buttons, but expand them.

12.35.5 Description

Array of Buttons. A ButtonArray is useful to have an array of buttons laid out vertically or horizontally. Only one button can be selected, and is referenced by its index in the array (first button is 0, second button is 1, etc.).

This is useful e.g. for joypad-friendly interfaces and option menus.

12.35.6 Member Function Description

• void add_button (String text, String tooltip="") | Append a new button to the array, with the specified text. |
• void add_icon_button (Texture icon, String text="", String tooltip="") | Append a new button to the array, with the specified icon and text. |
• void clear ( ) | |

12.35. ButtonArray
Remove all buttons from the array.

- `void erase_button(int button_idx)`

Remove the specified button in the array.

- `int get_button_count()` const

Return the amount of buttons in the array.

- `Texture get_button_icon(int button_idx)` const

Return the icon of the specified button.

- `String get_button_text(int button_idx)` const

Return the text of the specified button.

- `String get_button_tooltip(int button_idx)` const

- `int get_hovered()` const

Return the index of the currently hovered button in the array.

- `int get_selected()` const

Return the index of the currently selected button in the array.

- `bool is_flat()` const

- `void set_button_icon(int button_idx, Texture icon)`

Set the icon of the specified button.

- `void set_button_text(int button_idx, String text)`

Define the text of the specified button.

- `void set_button_tooltip(int button_idx, String text)`

- `void set_flat(bool enabled)`

- `void set_selected(int button_idx)`

Select a button in the array based on the given index.

### 12.36 ButtonGroup

**Inherits:** `BoxContainer < Container < Control < CanvasItem < Node < Object`

**Category:** Core

#### 12.36.1 Brief Description

Group of Buttons.
12.36.2 Member Functions

<table>
<thead>
<tr>
<th>Array</th>
<th>get_button_list ( ) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>BaseButton</td>
<td>getFocused_button ( ) const</td>
</tr>
<tr>
<td>int</td>
<td>get_pressed_button_index ( ) const</td>
</tr>
<tr>
<td>void</td>
<td>set_pressed_button ( BaseButton button )</td>
</tr>
</tbody>
</table>

12.36.3 Signals

- button_selected ( Object button )

12.36.4 Description

Group of Button. All direct and indirect children buttons become radios. Only one allows being pressed.

12.36.5 Member Function Description

- Array get_button_list ( ) const
  Return the list of all the buttons in the group.
  - BaseButton getFocused_button ( ) const
    Return the focused button.
  - BaseButton get_pressed_button ( ) const
    Return the pressed button.
  - int get_pressed_button_index ( ) const
    Return the index of the pressed button (by tree order).
  - void set_pressed_button ( BaseButton button )
    Set the button to be pressed.

12.37 Camera

Inherits: Spatial < Node < Object
Inherited By: InterpolatedCamera
Category: Core

12.37.1 Brief Description

Camera node, displays from a point of view.
12.37.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>clear_current()</td>
</tr>
<tr>
<td>Transform</td>
<td>get_camera_transform() const</td>
</tr>
<tr>
<td>Environment</td>
<td>get_environment() const</td>
</tr>
<tr>
<td>float</td>
<td>get_fov() const</td>
</tr>
<tr>
<td>float</td>
<td>get_h_offset() const</td>
</tr>
<tr>
<td>int</td>
<td>get_keep_aspect_mode() const</td>
</tr>
<tr>
<td>int</td>
<td>get_projection() const</td>
</tr>
<tr>
<td>float</td>
<td>get_size() const</td>
</tr>
<tr>
<td>float</td>
<td>get_v_offset() const</td>
</tr>
<tr>
<td>int</td>
<td>get_visible_layers() const</td>
</tr>
<tr>
<td>float</td>
<td>get_zfar() const</td>
</tr>
<tr>
<td>float</td>
<td>get_znear() const</td>
</tr>
<tr>
<td>bool</td>
<td>is_current() const</td>
</tr>
<tr>
<td>bool</td>
<td>is_position_behind(Vector3 world_point) const</td>
</tr>
<tr>
<td>void</td>
<td>make_current()</td>
</tr>
<tr>
<td>Vector3</td>
<td>project_local_ray_normal(Vector2 screen_point) const</td>
</tr>
<tr>
<td>Vector3</td>
<td>project_position(Vector2 screen_point) const</td>
</tr>
<tr>
<td>Vector3</td>
<td>project_ray_normal(Vector2 screen_point) const</td>
</tr>
<tr>
<td>Vector3</td>
<td>project_ray_origin(Vector2 screen_point) const</td>
</tr>
<tr>
<td>void</td>
<td>set_environment(Environment env)</td>
</tr>
<tr>
<td>void</td>
<td>set_h_offset(float ofs)</td>
</tr>
<tr>
<td>void</td>
<td>set_keep_aspect_mode(int mode)</td>
</tr>
<tr>
<td>void</td>
<td>set_orthogonal(float size, float z_near, float z_far)</td>
</tr>
<tr>
<td>void</td>
<td>set_perspective(float fov, float z_near, float z_far)</td>
</tr>
<tr>
<td>void</td>
<td>set_v_offset(float ofs)</td>
</tr>
<tr>
<td>void</td>
<td>set_visible_layers(int mask)</td>
</tr>
<tr>
<td>Vector2</td>
<td>unproject_position(Vector3 world_point) const</td>
</tr>
</tbody>
</table>

12.37.3 Numeric Constants

- **PROJECTION_PERSPECTIVE** = 0  — Perspective Projection (object’s size on the screen becomes smaller when far away).
- **PROJECTION_ORTHOGONAL** = 1  — Orthogonal Projection (objects remain the same size on the screen no matter how far away they are).
- **KEEP_WIDTH** = 0
- **KEEP_HEIGHT** = 1

12.37.4 Description

Camera is a special node that displays what is visible from its current location. Cameras register themselves in the nearest Viewport node (when ascending the tree). Only one camera can be active per viewport. If no viewport is available ascending the tree, the Camera will register in the global viewport. In other words, a Camera just provides 3D display capabilities to a Viewport, and, without one, a scene registered in that Viewport (or higher viewports) can’t be displayed.
12.37.5 Member Function Description

- **void clear_current ( )**
- **Transform get_camera_transform ( ) const**

Get the camera transform. Subclassed cameras (such as CharacterCamera) may provide different transforms than the Node transform.

- **Environment get_environment ( ) const**
- **float get_fov ( ) const**
- **float get_h_offset ( ) const**
- **int get_keep_aspect_mode ( ) const**
- **int get_projection ( ) const**
- **float get_size ( ) const**
- **float get_v_offset ( ) const**
- **int get_visible_layers ( ) const**
- **float get_zfar ( ) const**
- **float get_znear ( ) const**
- **bool is_current ( ) const**

Return whether the Camera is the current one in the Viewport, or plans to become current (if outside the scene tree).

- **bool is_position_behind ( Vector3 world_point ) const**
- **void make_current ( )**

Make this camera the current Camera for the Viewport (see class description). If the Camera Node is outside the scene tree, it will attempt to become current once it’s added.

- **Vector3 project_local_ray_normal ( Vector2 screen_point ) const**
- **Vector3 project_position ( Vector2 screen_point ) const**
- **Vector3 project_ray_normal ( Vector2 screen_point ) const**

Return a normal vector in worldspace, that is the result of projecting a point on the Viewport rectangle by the camera projection. This is useful for casting rays in the form of (origin,normal) for object intersection or picking.

- **Vector3 project_ray_origin ( Vector2 screen_point ) const**

Return a 3D position in worldspace, that is the result of projecting a point on the Viewport rectangle by the camera projection. This is useful for casting rays in the form of (origin,normal) for object intersection or picking.

- **void set_environment ( Environment env )**
- **void set_h_offset ( float ofs )**
- **void set_keep_aspect_mode ( int mode )**
- **void set_orthogonal ( float size, float z_near, float z_far )**

Set the camera projection to orthogonal mode, by specifying a width and the near and far clip planes in worldspace units. (As a hint, 2D games often use this projection, with values specified in pixels)

- **void set_perspective ( float fov, float z_near, float z_far )**

Set the camera projection to perspective mode, by specifying a FOV Y angle in degrees (FOV means Field of View), and the near and far clip planes in worldspace units.
- void set_v_offset (float ofs)
- void set_visible_layers (int mask)
- Vector2 unproject_position (Vector3 world_point) const

Return how a 3D point in worldspace maps to a 2D coordinate in the Viewport rectangle.

### 12.38 Camera2D

**Inherits:** Node2D < CanvasItem < Node < Object  
**Category:** Core

#### 12.38.1 Brief Description

Camera node for 2D scenes.

#### 12.38.2 Member Functions

<table>
<thead>
<tr>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>void align ( )</td>
</tr>
<tr>
<td>void clear_current ( )</td>
</tr>
<tr>
<td>void force_update_scroll ( )</td>
</tr>
<tr>
<td>int get_anchor_mode ( ) const</td>
</tr>
<tr>
<td>Vector2 get_camera_pos ( ) const</td>
</tr>
<tr>
<td>Vector2 get_camera_screen_center ( ) const</td>
</tr>
<tr>
<td>Viewport get_custom_viewport ( ) const</td>
</tr>
<tr>
<td>float get_drag_margin (int margin) const</td>
</tr>
<tr>
<td>float get_follow_smoothing ( ) const</td>
</tr>
<tr>
<td>float get_h_offset ( ) const</td>
</tr>
<tr>
<td>int get_limit (int margin) const</td>
</tr>
<tr>
<td>Vector2 get_offset ( ) const</td>
</tr>
<tr>
<td>float get_v_offset ( ) const</td>
</tr>
<tr>
<td>float get_zoom ( ) const</td>
</tr>
<tr>
<td>bool is_current ( ) const</td>
</tr>
<tr>
<td>bool is_follow_smoothing_enabled ( ) const</td>
</tr>
<tr>
<td>bool is_h_drag_enabled ( ) const</td>
</tr>
<tr>
<td>bool is_rotating ( ) const</td>
</tr>
<tr>
<td>bool is_v_drag_enabled ( ) const</td>
</tr>
<tr>
<td>void make_current ( )</td>
</tr>
<tr>
<td>void reset_smoothing ( )</td>
</tr>
<tr>
<td>void set_anchor_mode (int anchor_mode)</td>
</tr>
<tr>
<td>void set_custom_viewport (Viewport viewport)</td>
</tr>
<tr>
<td>void set_drag_margin (int margin, float drag_margin)</td>
</tr>
<tr>
<td>void set_enable_follow_smoothing (bool follow_smoothing)</td>
</tr>
<tr>
<td>void set_follow_smoothing (float follow_smoothing)</td>
</tr>
<tr>
<td>void set_h_drag_enabled (bool enabled)</td>
</tr>
<tr>
<td>void set_h_offset (float ofs)</td>
</tr>
<tr>
<td>void set_limit (int margin, int limit)</td>
</tr>
<tr>
<td>void set_offset (Vector2 offset)</td>
</tr>
</tbody>
</table>

Continued on next page
void set_rotating ( bool rotating )
void set_v_drag_enabled ( bool enabled )
void set_v_offset ( float ofs )
void set_zoom ( Vector2 zoom )

12.38.3 Numeric Constants

- ANCHOR_MODE_DRAG_CENTER = 1
- ANCHOR_MODE_FIXED_TOP_LEFT = 0

12.38.4 Description

Camera node for 2D scenes. It forces the screen (current layer) to scroll following this node. This makes it easier (and faster) to program scrollable scenes than manually changing the position of CanvasItem based nodes.

This node is intended to be a simple helper get things going quickly and it may happen often that more functionality is desired to change how the camera works. To make your own custom camera node, simply inherit from Node2D and change the transform of the canvas by calling get_viewport().set_canvas_transform(m) in Viewport.

12.38.5 Member Function Description

- void align ( )
  Align the camera to the tracked node

- void clear_current ( )
- void force_update_scroll ( )
  Force the camera to update scroll immediately.

- int get_anchor_mode ( ) const
- Vector2 get_camera_pos ( ) const
  Return the camera position.

- Vector2 get_camera_screen_center ( ) const
- Viewport get_custom_viewport ( ) const
  Return the viewport RID for this layer.

- float get_drag_margin ( int margin ) const
  Return the margins needed to drag the camera (see set_drag_margin).

- float get_follow_smoothing ( ) const
- float get_h_offset ( ) const
- int get_limit ( int margin ) const
  Return the scrolling limit in pixels.

- Vector2 get_offset ( ) const
  Return the scroll offset.

- float get_v_offset ( ) const
• `Vector2 get_zoom ( ) const
• `bool is_current ( ) const

Return true of this is the current camera (see `make_current`).

• `bool is_follow_smoothing_enabled ( ) const
• `bool is_h_drag_enabled ( ) const
• `bool is_rotating ( ) const
• `bool is_v_drag_enabled ( ) const
• `void make_current ( )

Make this the current 2D camera for the scene (viewport and layer), in case there’s many cameras in the scene.

• `void reset_smoothing ( )

Set the camera’s position immediately to its current smoothing destination.

This has no effect if smoothing is disabled.

• `void set_anchor_mode ( `int anchor_mode )
• `void set_custom_viewport ( `Viewport viewport )
• `void set_drag_margin ( `int margin, `float drag_margin )

Set the margins needed to drag the camera (relative to the screen size). Margin uses the MARGIN_* enum. Drag margins of 0,0,0,0 will keep the camera at the center of the screen, while drag margins of 1,1,1,1 will only move when the camera is at the edges.

• `void set_enable_follow_smoothing ( `bool follow_smoothing )
• `void set_follow_smoothing ( `float follow_smoothing )
• `void set_h_drag_enabled ( `bool enabled )
• `void set_h_offset ( `float ofs )
• `void set_limit ( `int margin, `int limit )

Set the scrolling limit in pixels.

• `void set_offset ( `Vector2 offset )

Set the scroll offset. Useful for looking around or camera shake animations.

• `void set_rotating ( `bool rotating )
• `void set_v_drag_enabled ( `bool enabled )
• `void set_v_offset ( `float ofs )
• `void set_zoom ( `Vector2 zoom )

12.39 CanvasItem

Inherits: `Node < `Object

Inherited By: `Node2D, `Control

Category: Core
12.39.1 Brief Description

Base class of anything 2D.

12.39.2 Member Functions

```csharp
void _draw() virtual
float draw_char(Font font, Vector2 pos, String char, String next, Color modulate=Color(1,1,1,1))
void draw_circle(Vector2 pos, float radius, Color color)
void draw_colored_polygon(Vector2Array points, Color color, Vector2Array uvs=Vector2Array([]), Texture texture=NULL)
void draw_line(Vector2 from, Vector2 to, Color color, float width=1)
void draw_polygon(Vector2Array points, ColorArray colors, Vector2Array uvs=Vector2Array([]), Texture texture=NULL)
void draw_primitive(Vector2Array points, ColorArray colors, Vector2Array uvs, Texture texture=NULL, float width=1)
void draw_rect(Rect2 rect, Color color)
void draw_set_transform(Vector2 pos, float rot, Vector2 scale)
void draw_set_transform_matrix(Matrix32 xform)
void draw_string(Font font, Vector2 pos, String text, Color modulate=Color(1,1,1,1), int clip_w=-1)
void draw_style_box(StyleBox style_box, Rect2 rect)
void draw_texture(Texture texture, Vector2 pos, Color modulate=Color(1,1,1,1))
void draw_texture_rect(Texture texture, Rect2 rect, bool tile, Color modulate=Color(1,1,1,1), bool transpose=false)
void draw_texture_rect_region(Texture texture, Rect2 rect, Rect2 src_rect, Color modulate=Color(1,1,1,1), bool transpose=false)
Variant edit_get_state() const
void edit_rotate(float degrees)
void edit_set_rect(Rect2 rect)
void edit_set_state(Variant state)
int get_blend_mode() const
RID get_canvas() const
RID get_canvas_item() const
Matrix32 get_canvas_transform() const
Vector2 get_global_mouse_pos() const
Matrix32 get_global_transform() const
Matrix32 get_global_transform_with_canvas() const
Rect2 get_item_and_children_rect() const
Rect2 get_item_rect() const
int get_light_mask() const
Vector2 get_local_mouse_pos() const
CanvasItemMaterial get_material() const
float get_opacity() const
float get_self_opacity() const
Matrix32 get_transform() const
bool get_use_parent_material() const
Rect2 getViewportRect() const
Matrix32 getViewportTransform() const
Object get_world_2d() const
void hide()
bool is_draw_behind_parent_enabled() const
bool is_hidden() const
bool is_set_as_toplevel() const
bool is_visible() const
Vector2 make_canvas_pos_local(Vector2 screen_point) const
```
Table 12.11 – continued from previous page

<table>
<thead>
<tr>
<th>InputEvent</th>
<th>make_input_local (InputEvent event) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>set_as_toplevel (bool enable)</td>
</tr>
<tr>
<td>void</td>
<td>set_blend_mode (int blend_mode)</td>
</tr>
<tr>
<td>void</td>
<td>set_draw_behind_parent (bool enable)</td>
</tr>
<tr>
<td>void</td>
<td>set_hidden (bool hidden)</td>
</tr>
<tr>
<td>void</td>
<td>set_light_mask (int light_mask)</td>
</tr>
<tr>
<td>void</td>
<td>set_material (CanvasItemMaterial material)</td>
</tr>
<tr>
<td>void</td>
<td>set_opacity (float opacity)</td>
</tr>
<tr>
<td>void</td>
<td>set_self_opacity (float self_opacity)</td>
</tr>
<tr>
<td>void</td>
<td>set_use_parent_material (bool enable)</td>
</tr>
<tr>
<td>void</td>
<td>show ( )</td>
</tr>
<tr>
<td>void</td>
<td>update ( )</td>
</tr>
</tbody>
</table>

12.39.3 Signals

- **draw ( )**
  Emitted when the CanvasItem must redraw. This can only be connected realtime, as deferred will not allow drawing.

- **hide ( )**
  Emitted when becoming hidden.

- **item_rect_changed ( )**
  Emitted when the item rect has changed.

- **visibility_changed ( )**
  Emitted when the visibility (hidden/visible) changes.

12.39.4 Numeric Constants

- **BLEND_MODE_MIX = 0** — Mix blending mode. Colors are assumed to be independent of the alpha (opacity) value.
- **BLEND_MODE_ADD = 1** — Additive blending mode.
- **BLEND_MODE_SUB = 2** — Subtractive blending mode.
- **BLEND_MODE_MUL = 3** — Multiplicative blending mode.
- **BLEND_MODE_PREMULT_ALPHA = 4** — Mix blending mode. Colors are assumed to be premultiplied by the alpha (opacity) value.
- **NOTIFICATION_DRAW = 30** — CanvasItem is requested to draw.
- **NOTIFICATION_VISIBILITY_CHANGED = 31** — Canvas item visibility has changed.
- **NOTIFICATION_ENTER_CANVAS = 32** — Canvas item has entered the canvas.
- **NOTIFICATION_EXIT_CANVAS = 33** — Canvas item has exited the canvas.
- **NOTIFICATION_TRANSFORM_CHANGED = 29** — Canvas item transform has changed. Only received if requested.
12.39.5 Description

Base class of anything 2D. Canvas items are laid out in a tree and children inherit and extend the transform of their parent. CanvasItem is extended by Control, for anything GUI related, and by Node2D for anything 2D engine related.

Any CanvasItem can draw. For this, the “update” function must be called, then NOTIFICATION_DRAW will be received on idle time to request redraw. Because of this, canvas items don’t need to be redraw on every frame, improving the performance significantly. Several functions for drawing on the CanvasItem are provided (see draw_* functions). They can only be used inside the notification, signal or _draw() overrides function, though.

Canvas items are draw in tree order. By default, children are on top of their parents so a root CanvasItem will be drawn behind everything (this can be changed per item though).

Canvas items can also be hidden (hiding also their subtree). They provide many means for changing standard parameters such as opacity (for it and the subtree) and self opacity, blend mode.

Ultimately, a transform notification can be requested, which will notify the node that its global position changed in case the parent tree changed.

12.39.6 Member Function Description

• void _draw ( ) virtual

   Called (if exists) to draw the canvas item.

   • float draw_char ( Font font, Vector2 pos, String char, String next, Color modulate=Color(1,1,1,1) )

   Draw a string character using a custom font. Returns the advance, depending on the char width and kerning with an optional next char.

   • void draw_circle ( Vector2 pos, float radius, Color color )

   Draw a colored circle.

   • void draw_colored_polygon ( Vector2Array points, Color color, Vector2Array uvs=Vector2Array([]), Texture texture=NULL )

   Draw a colored polygon of any amount of points, convex or concave.

   • void draw_line ( Vector2 from, Vector2 to, Color color, float width=1 )

   Draw a line from a 2D point to another, with a given color and width.

   • void draw_polygon ( Vector2Array points, ColorArray colors, Vector2Array uvs=Vector2Array([]), Texture texture=NULL )

   Draw a polygon of any amount of points, convex or concave.

   • void draw_primitive ( Vector2Array points, ColorArray colors, Vector2Array uvs, Texture texture=NULL, float width=1 )

   Draw a custom primitive, 1 point for a point, 2 points for a line, 3 points for a triangle and 4 points for a quad.

   • void draw_rect ( Rect2 rect, Color color )

   Draw a colored rectangle.

   • void draw_set_transform ( Vector2 pos, float rot, Vector2 scale )

   Set a custom transform for drawing. Anything drawn afterwards will be transformed by this.

   • void draw_set_transform_matrix ( Matrix32 xform )

   • void draw_string ( Font font, Vector2 pos, String text, Color modulate=Color(1,1,1,1), int clip_w=-1 )
Draw a string using a custom font.

- void **draw_style_box** ( *StyleBox* style_box, *Rect2* rect )

Draw a styled rectangle.

- void **draw_texture** ( *Texture* texture, *Vector2* pos, *Color* modulate=Color(1,1,1,1) )

Draw a texture at a given position.

- void **draw_texture_rect** ( *Texture* texture, *Rect2* rect, *Color* modulate=Color(1,1,1,1), *bool* transpose=false )

Draw a textured rectangle at a given position, optionally modulated by a color. Transpose swaps the x and y coordinates when reading the texture.

- void **draw_texture_rect_region** ( *Texture* texture, *Rect2* rect, *Rect2* src_rect, *Color* modulate=Color(1,1,1,1), *bool* transpose=false )

Draw a textured rectangle region at a given position, optionally modulated by a color. Transpose swaps the x and y coordinates when reading the texture.

- **Variant** **edit_get_state** () **const**

Used for editing, returns an opaque value representing the transform state.

- void **edit_rotate** ( *float* degrees )

Used for editing, handle rotation.

- void **edit_set_rect** ( *Rect2* rect )

- void **edit_set_state** ( *Variant* state )

Set the transform state of this CanvasItem. For *Node2D*, this is an *Array* with (in order) a *Vector2* for position, a float for rotation and another *Vector2* for scale. For *Control* this is a *Rect2* with the position and size.

- **int** **get_blend_mode** () **const**

Return the current blending mode from enum BLEND_MODE_*_.

- **RID** **get_canvas** () **const**

Return the **RID** of the **World2D** canvas where this item is in.

- **RID** **get_canvas_item** () **const**

Return the canvas item **RID** used by **VisualServer** for this item.

- **Matrix32** **get_canvas_transform** () **const**

Get the transform matrix of this item’s canvas.

- **Vector2** **get_global_mouse_pos** () **const**

Get the global position of the mouse.

- **Matrix32** **get_global_transform** () **const**

Get the global transform matrix of this item.

- **Matrix32** **get_global_transform_with_canvas** () **const**

Get the global transform matrix of this item in relation to the canvas.

- **Rect2** **get_item_and_children_rect** () **const**

Get a *Rect2* with the boundaries of this item and its children.

- **Rect2** **get_item_rect** () **const**
Return a rect containing the editable boundaries of the item.

- **int get_light_mask ( ) const**
  Get this item’s light mask number.

- **Vector2 get_local_mouse_pos ( ) const**
  Get the mouse position relative to this item’s position.

- **CanvasItemMaterial get_material ( ) const**
  Get the material of this item.

- **float get_opacity ( ) const**
  Return the canvas item opacity. This affects the canvas item and all the children.

- **float get_self_opacity ( ) const**
  Return the canvas item self-opacity.

- **Matrix32 get_transform ( ) const**
  Get the transform matrix of this item.

- **bool get_use_parent_material ( ) const**
  Get whether this item uses its parent’s material.

- **Rect2 get_viewport_rect ( ) const**
  Get the viewport’s boundaries as a Rect2.

- **Matrix32 get_viewport_transform ( ) const**
  Get this item’s transform in relation to the viewport.

- **Object get_world_2d ( ) const**
  Get the World2D where this item is in.

- **void hide ( )**
  Hide the CanvasItem currently visible.

- **bool is_draw_behind_parent_enabled ( ) const**
  Return whether the item is drawn behind its parent.

- **bool is_hidden ( ) const**
  Return true if this CanvasItem is hidden. Note that the CanvasItem may not be visible, but as long as it’s not hidden (hide called) the function will return false.

- **bool is_set_as_toplevel ( ) const**
  Return if set as toplevel. See set_as_toplevel.

- **bool is_visible ( ) const**
  Return true if this CanvasItem is visible. It may be invisible because itself or a parent canvas item is hidden.

- **Vector2 make_canvas_pos_local ( Vector2 screen_point ) const**

- **InputEvent make_input_local ( InputEvent event ) const**
  Takes a global input event and convert to this item’s coordinate system.

- **void set_as_toplevel ( bool enable )**
Set as top level. This means that it will not inherit transform from parent canvas items.

- void set_blend_mode (int blend_mode)
  Set the blending mode from enum BLEND_MODE_*.

- void set_draw_behind_parent (bool enable)
  Set whether the canvas item is drawn behind its parent.

- void set_hidden (bool hidden)
  Set whether this item should be hidden or not. Note that no matter what is set here this item won’t be shown if its parent or grandparents nodes are also hidden. A hidden CanvasItem make all children hidden too.

- void set_light_mask (int light_mask)
  Set the light mask number of this item.

- void set_material (CanvasItemMaterial material)
  Set the material of this item.

- void set_opacity (float opacity)
  Set canvas item opacity. This will affect the canvas item and all the children.

- void set_self_opacity (float self_opacity)
  Set canvas item self-opacity. This does not affect the opacity of children items.

- void set_use_parent_material (bool enable)
  Set whether or not this item should use its parent’s material.

- void show ()
  Show the CanvasItem currently hidden.

- void update ()
  Queue the CanvasItem for update. NOTIFICATION_DRAW will be called on idle time to request redraw.

### 12.40 CanvasItemMaterial

**Inherits:** Resource < Reference < Object

**Category:** Core

#### 12.40.1 Brief Description

#### 12.40.2 Member Functions

<table>
<thead>
<tr>
<th>Shader</th>
<th>get_shader ( ) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>get_shader_param (String param ) const</td>
</tr>
<tr>
<td>int</td>
<td>get_shading_mode ( ) const</td>
</tr>
<tr>
<td>void</td>
<td>set_shader (Shader shader)</td>
</tr>
<tr>
<td>void</td>
<td>set_shader_param (String param, Variant value )</td>
</tr>
<tr>
<td>void</td>
<td>set_shading_mode (int mode)</td>
</tr>
</tbody>
</table>
12.40.3 Numeric Constants

• SHADING_NORMAL = 0
• SHADING_UNSHADED = 1
• SHADING_ONLY_LIGHT = 2

12.40.4 Member Function Description

• Shader get_shader ( ) const
• void get_shader_param ( String param ) const
• int get_shading_mode ( ) const
• void set_shader ( Shader shader )
• void set_shader_param ( String param, Variant value )
• void set_shading_mode ( int mode )

12.41 CanvasItemShader

Inherits: Shader < Resource < Reference < Object
Category: Core

12.41.1 Brief Description

12.42 CanvasItemShaderGraph

Inherits: ShaderGraph < Shader < Resource < Reference < Object
Category: Core

12.42.1 Brief Description

12.43 CanvasLayer

Inherits: Node < Object
Inherited By: ParallaxBackground
Category: Core

12.43.1 Brief Description

Canvas Item layer.
## 12.43.2 Member Functions

<table>
<thead>
<tr>
<th>Class</th>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Viewport</td>
<td>get_custom_viewport() const</td>
<td>Return the viewport RID for this layer.</td>
</tr>
<tr>
<td>int</td>
<td>get_layer() const</td>
<td>Return the layer index, determines the draw order, a lower value will be below a higher one.</td>
</tr>
<tr>
<td>Vector2</td>
<td>get_offset() const</td>
<td>Return the base offset for this layer (helper).</td>
</tr>
<tr>
<td>float</td>
<td>get_rotation() const</td>
<td>Return the base rotation for this layer (helper).</td>
</tr>
<tr>
<td>float</td>
<td>get_rotationd() const</td>
<td>Get rotation of the layer in degree.</td>
</tr>
<tr>
<td>Vector2</td>
<td>get_scale() const</td>
<td>Return the base scale for this layer (helper).</td>
</tr>
<tr>
<td>Matrix32</td>
<td>get_transform() const</td>
<td>Return the base transform for this layer.</td>
</tr>
<tr>
<td>World2D</td>
<td>get_world_2d() const</td>
<td>Return the World2D used by this layer.</td>
</tr>
<tr>
<td>void</td>
<td>set_custom_viewport(Viewport viewport) void</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>set_layer(int layer) void</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>set_offset(Vector2 offset) void</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>set_rotation(float radians) void</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>set_rotationd(float degrees) void</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>set_scale(Vector2 scale) void</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>set_transform(Matrix32 transform) void</td>
<td></td>
</tr>
</tbody>
</table>

### 12.43.3 Description

Canvas Item layer. CanvasItem nodes that are direct or indirect children of a CanvasLayer will be drawn in that layer. The layer is a numeric index that defines the draw order. The default 2D scene renders with index 0, so a CanvasLayer with index -1 will be drawn below, and one with index 1 will be drawn above. This is very useful for HUDs (in layer 1+ or above), or backgrounds (in layer -1 or below).

### 12.43.4 Member Function Description

- **Viewport** get_custom_viewport() const
  Return the viewport RID for this layer.

- **int** get_layer() const
  Return the layer index, determines the draw order, a lower value will be below a higher one.

- **Vector2** get_offset() const
  Return the base offset for this layer (helper).

- **float** get_rotation() const
  Return the base rotation for this layer (helper).

- **float** get_rotationd() const
  Get rotation of the layer in degree.

- **Vector2** get_scale() const
  Return the base scale for this layer (helper).

- **Matrix32** get_transform() const
  Return the base transform for this layer.

- **World2D** get_world_2d() const
  Return the World2D used by this layer.
• void set_custom_viewport (Viewport viewport)
• void set_layer (int layer)
Set the layer index, determines the draw order, a lower value will be below a higher one.
• void set_offset (Vector2 offset)
Set the base offset for this layer (helper).
• void set_rotation (float radians)
Set the base rotation for this layer (helper).
• void set_rotationd (float degrees)
Set rotation of the layer in degree.
• void set_scale (Vector2 scale)
Set the base scale for this layer (helper).
• void set_transform (Matrix32 transform)
Set the base transform for this layer.

12.44 CanvasModulate

Inherits: Node2D < CanvasItem < Node < Object
Category: Core

12.44.1 Brief Description

Tint the entire canvas

12.44.2 Member Functions

<table>
<thead>
<tr>
<th>Color</th>
<th>get_color ( ) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>set_color ( Color color )</td>
</tr>
</tbody>
</table>

12.44.3 Description

CanvasModulate tints the canvas elements using its asigned color

12.44.4 Member Function Description

• Color get_color ( ) const
  Gets the canvas tint color
• void set_color ( Color color )
  Sets the canvas tint color
12.45 CapsuleShape

Inherits: Shape < Resource < Reference < Object

Category: Core

12.45.1 Brief Description

Capsule shape resource.

12.45.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>float</td>
<td>get_height() const</td>
</tr>
<tr>
<td>float</td>
<td>get_radius() const</td>
</tr>
<tr>
<td>void</td>
<td>set_height(float height)</td>
</tr>
<tr>
<td>void</td>
<td>set_radius(float radius)</td>
</tr>
</tbody>
</table>

12.45.3 Description

Capsule shape resource, which can be set into a PhysicsBody or area.

12.45.4 Member Function Description

- float get_height() const
  
  Return the capsule height.

- float get_radius() const
  
  Return the capsule radius.

- void set_height(float height)
  
  Set the capsule height.

- void set_radius(float radius)
  
  Set the capsule radius.

12.46 CapsuleShape2D

Inherits: Shape2D < Resource < Reference < Object

Category: Core

12.46.1 Brief Description

Capsule 2D shape resource for physics.
12.46.2 Member Functions

<table>
<thead>
<tr>
<th>float</th>
<th>get_height ( ) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>float</td>
<td>get_radius ( ) const</td>
</tr>
<tr>
<td>void</td>
<td>set_height ( float height )</td>
</tr>
<tr>
<td>void</td>
<td>set_radius ( float radius )</td>
</tr>
</tbody>
</table>

12.46.3 Description

Capsule 2D shape resource for physics. A capsule (or sometimes called “pill”) is like a line grown in all directions. It has a radius and a height, and is often useful for modeling biped characters.

12.46.4 Member Function Description

- `float get_height ( ) const`
  Return the height of the `CapsuleShape2D`.
- `float get_radius ( ) const`
  Return the radius of the `CapsuleShape2D`.
- `void set_height ( float height )`
  Set the height of the `CapsuleShape2D`.
- `void set_radius ( float radius )`
  Set the radius of the `CapsuleShape2D`.

12.47 CenterContainer

Inherits: `Container < Control < CanvasItem < Node < Object`

Category: Core

12.47.1 Brief Description

Keeps children controls centered.

12.47.2 Member Functions

<table>
<thead>
<tr>
<th>bool</th>
<th>is_using_top_left ( ) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>set_use_top_left ( bool enable )</td>
</tr>
</tbody>
</table>

12.47.3 Description

CenterContainer Keeps children controls centered. This container keeps all children to their minimum size, in the center.
12.47.4 Member Function Description

- 

  bool is_using_top_left ( ) const

  Should put children to the top left corner instead of center of the container.

- 

  void set_use_top_left ( bool enable )

  This function will anchor the container children to the top left corner of the container boundaries, moving all its children to that position, (the children new center will be the top left corner of the container).

12.48 CheckBox

Inherits: Button < BaseButton < Control < CanvasItem < Node < Object

Category: Core

12.48.1 Brief Description

Binary choice user interface widget

12.48.2 Description

A checkbox allows the user to make a binary choice (choosing only one of two possible options), for example Answer ‘yes’ or ‘no’.

12.49 CheckButton

Inherits: Button < BaseButton < Control < CanvasItem < Node < Object

Category: Core

12.49.1 Brief Description

Checkable button.

12.49.2 Description

CheckButton is a toggle button displayed as a check field.

12.50 CircleShape2D

Inherits: Shape2D < Resource < Reference < Object

Category: Core
12.50.1 Brief Description

Circular Shape for 2D Physics.

12.50.2 Member Functions

<table>
<thead>
<tr>
<th>float</th>
<th>get_radius ( ) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>set_radius ( float radius )</td>
</tr>
</tbody>
</table>

12.50.3 Description

Circular Shape for 2D Physics. This shape is useful for modeling balls or small characters and its collision detection with everything else is very fast.

12.50.4 Member Function Description

- float get_radius ( ) const
  Return the radius of the circle shape.
- void set_radius ( float radius )
  Set the radius of the circle shape.

12.51 CollisionObject

Inherits: Spatial < Node < Object
Inherited By: PhysicsBody, Area
Category: Core
12.51.1 Brief Description

12.51.2 Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void <code>_input_event</code> (Object camera, InputEvent event, Vector3 click_pos, Vector3 click_normal, int shape_idx) virtual</td>
<td>virtual function takes <code>Object camera</code>, <code>InputEvent event</code>, <code>Vector3 click_pos</code>, <code>Vector3 click_normal</code>, and <code>int shape_idx</code> as arguments.</td>
</tr>
<tr>
<td>void <code>add_shape</code> (Shape shape, Transform transform=Transform())</td>
<td>Adds a shape to the scene.</td>
</tr>
<tr>
<td>void <code>clear_shapes</code> ()</td>
<td>Removes all shapes from the scene.</td>
</tr>
<tr>
<td>bool <code>get_capture_input_on_drag</code> () const</td>
<td>Returns a boolean indicating whether capture input is enabled on drag.</td>
</tr>
<tr>
<td>RID <code>get_rid</code> () const</td>
<td>Returns the RID of the object.</td>
</tr>
<tr>
<td>Shape <code>get_shape</code> (int shape_idx) const</td>
<td>Returns the shape at the specified index.</td>
</tr>
<tr>
<td>int <code>get_shape_count</code> () const</td>
<td>Returns the count of shapes in the scene.</td>
</tr>
<tr>
<td>Transform <code>get_shape_transform</code> (int shape_idx) const</td>
<td>Returns the transform of the specified shape.</td>
</tr>
<tr>
<td>bool <code>is_ray_pickable</code> () const</td>
<td>Returns a boolean indicating whether the object is ray pickable.</td>
</tr>
<tr>
<td>bool <code>is_shape_set_as_trigger</code> (int shape_idx) const</td>
<td>Returns a boolean indicating whether the specified shape is set as a trigger.</td>
</tr>
<tr>
<td>void <code>remove_shape</code> (int shape_idx)</td>
<td>Removes the specified shape from the scene.</td>
</tr>
<tr>
<td>void <code>set_capture_input_on_drag</code> (bool enable)</td>
<td>Enables or disables capture input on drag.</td>
</tr>
<tr>
<td>void <code>set_shape</code> (int shape_idx, Shape shape)</td>
<td>Sets the specified shape at the specified index.</td>
</tr>
<tr>
<td>void <code>set_shape_as_trigger</code> (int shape_idx, bool enable)</td>
<td>Sets the specified shape as a trigger with the specified enable state.</td>
</tr>
<tr>
<td>void <code>set_shape_transform</code> (int shape_idx, Transform transform)</td>
<td>Sets the transform of the specified shape.</td>
</tr>
</tbody>
</table>

12.51.3 Signals

- `input_event` (Object camera, InputEvent event, Vector3 click_pos, Vector3 click_normal, int shape_idx)
- `mouse_enter`()
- `mouse_exit`()

12.51.4 Member Function Description

- void `_input_event` (Object camera, InputEvent event, Vector3 click_pos, Vector3 click_normal, int shape_idx) virtual
- void `add_shape` (Shape shape, Transform transform=Transform())
- void `clear_shapes`()
- bool `get_capture_input_on_drag` () const
- RID `get_rid` () const
- Shape `get_shape` (int shape_idx) const
- int `get_shape_count` () const
- Transform `get_shape_transform` (int shape_idx) const
- bool `is_ray_pickable` () const
- bool `is_shape_set_as_trigger` (int shape_idx) const
- void `remove_shape` (int shape_idx)
- void `set_capture_input_on_drag` (bool enable)
• void set_ray_pickable ( bool ray_pickable )
• void set_shape ( int shape_idx, Shape shape )
• void set_shape_as_trigger ( int shape_idx, bool enable )
• void set_shape_transform ( int shape_idx, Transform transform )

12.52 CollisionObject2D

Inherits: Node2D < CanvasItem < Node < Object
Inherited By: Area2D, PhysicsBody2D
Category: Core

12.52.1 Brief Description

Base node for 2D collisionables.

12.52.2 Member Functions

<table>
<thead>
<tr>
<th>void</th>
<th>_input_event ( Object viewport, InputEvent event, int shape_idx ) virtual</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>add_shape ( Shape2D shape, Matrix32 transform=((1, 0), (0, 1), (0, 0)) )</td>
</tr>
<tr>
<td>void</td>
<td>clear_shapes ( )</td>
</tr>
<tr>
<td>RID</td>
<td>get_rid ( ) const</td>
</tr>
<tr>
<td>Shape2D</td>
<td>get_shape ( int shape_idx ) const</td>
</tr>
<tr>
<td>int</td>
<td>get_shape_count ( ) const</td>
</tr>
<tr>
<td>Matrix32</td>
<td>get_shape_transform ( int shape_idx ) const</td>
</tr>
<tr>
<td>bool</td>
<td>is_pickable ( ) const</td>
</tr>
<tr>
<td>bool</td>
<td>is_shape_set_as_trigger ( int shape_idx ) const</td>
</tr>
<tr>
<td>void</td>
<td>remove_shape ( int shape_idx )</td>
</tr>
<tr>
<td>void</td>
<td>set_pickable ( bool enabled )</td>
</tr>
<tr>
<td>void</td>
<td>set_shape ( int shape_idx, Shape shape )</td>
</tr>
<tr>
<td>void</td>
<td>set_shape_as_trigger ( int shape_idx, bool enable )</td>
</tr>
<tr>
<td>void</td>
<td>set_shape_transform ( int shape_idx, Matrix32 transform )</td>
</tr>
</tbody>
</table>

12.52.3 Signals

• input_event ( Object viewport, InputEvent event, int shape_idx )

This signal triggers when an input event fires over a shape. The first parameter is the viewport where the event took place. The second holds the input event received, and the third the shape of this object where it happened.

• mouse_enter ( )

This event fires only once when the mouse pointer enters any shape of this object.

• mouse_exit ( )

This event fires only once when the mouse pointer exits all shapes of this object.
12.52.4 Description

CollisionObject2D is the base class for 2D physics collisionables. They can hold any number of 2D collision shapes. Usually, they are edited by placing CollisionShape2D and/or CollisionPolygon2D nodes as children. Such nodes are for reference and not present outside the editor, so code should use the regular shape API.

12.52.5 Member Function Description

- **void** _input_event ( Object viewport, InputEvent event, int shape_idx ) virtual
  
  This method can be used to override normal input processing. The first parameter is the viewport where the event took place. The second holds the input event received, and the third the shape of this object where it happened.

- **void** add_shape ( Shape2D shape, Matrix32 transform=((1, 0), (0, 1), (0, 0)) )
  
  Add a Shape2D to the collision body, with a given custom transform.

- **void** clear_shapes ( )
  
  Remove all shapes.

- **RID** get_rid ( ) const
  
  Return the RID of this object.

- **Shape2D** get_shape ( int shape_idx ) const
  
  Return the shape in the given index.

- **int** get_shape_count ( ) const
  
  Return the amount of shapes in the collision body. Because a CollisionPolygon2D can generate more than one Shape2D, the amount returned does not have to match the sum of CollisionShape2D and CollisionPolygon2D.

- **Matrix32** get_shape_transform ( int shape_idx ) const
  
  Return the shape transform in the given index.

- **bool** is_pickable ( ) const
  
  Return whether this object is pickable.

- **bool** is_shape_set_as_trigger ( int shape_idx ) const
  
  Return whether a shape is a trigger. A trigger shape detects collisions, but is otherwise unaffected by physics (i.e. colliding objects will not get blocked).

- **void** remove_shape ( int shape_idx )
  
  Remove the shape in the given index.

- **void** set_pickable ( bool enabled )
  
  Set whether this object is pickable. A pickable object can detect the mouse pointer enter/leave it and, if the mouse is inside it, report input events.

- **void** set_shape ( int shape_idx, Shape shape )
  
  Change a shape in the collision body.

- **void** set_shape_as_trigger ( int shape_idx, bool enable )
  
  Set whether a shape is a trigger. A trigger shape detects collisions, but is otherwise unaffected by physics (i.e. colliding objects will not get blocked).

- **void** set_shape_transform ( int shape_idx, Matrix32 transform )
  
  Set the shape transform.
Change the shape transform in the collision body.

### 12.53 CollisionPolygon

**Inherits:** Spatial < Node < Object  
**Category:** Core

#### 12.53.1 Brief Description

#### 12.53.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>get_build_mode() const</td>
</tr>
<tr>
<td>int</td>
<td>get_collision_object_first_shape() const</td>
</tr>
<tr>
<td>int</td>
<td>get_collision_object_last_shape() const</td>
</tr>
<tr>
<td>float</td>
<td>get_depth() const</td>
</tr>
<tr>
<td>Vector2Array</td>
<td>get_polygon() const</td>
</tr>
<tr>
<td>void</td>
<td>set_build_mode(int build_mode)</td>
</tr>
<tr>
<td>void</td>
<td>set_depth(float depth)</td>
</tr>
<tr>
<td>void</td>
<td>set_polygon(Vector2Array polygon)</td>
</tr>
</tbody>
</table>

#### 12.53.3 Member Function Description

- int get_build_mode() const
- int get_collision_object_first_shape() const
- int get_collision_object_last_shape() const
- float get_depth() const
- Vector2Array get_polygon() const
- void set_build_mode(int build_mode)
- void set_depth(float depth)
- void set_polygon(Vector2Array polygon)

### 12.54 CollisionPolygon2D

**Inherits:** Node2D < CanvasItem < Node < Object  
**Category:** Core

#### 12.54.1 Brief Description

Editor-only class for easy editing of collision polygons.
12.54.3 Description

Editor-only class. This is not present when running the game. It’s used in the editor to properly edit and position collision shapes in `CollisionObject2D`. This is not accessible from regular code. This class is for editing custom shape polygons.

12.54.4 Member Function Description

- **int get_build_mode ( ) const**
  
  Return whether the polygon is a `ConvexPolygonShape2D` (build_mode==0), or a `ConcavePolygonShape2D` (build_mode==1).

- **int get_collision_object_first_shape ( ) const**
  
  Return the index of the first shape generated by the editor.

  When `build_mode` is set to generate convex polygons, the shape shown in the editor may be decomposed into many convex polygons. In that case, a range of indexes is needed to directly access the `Shape2D`.

  When `build_mode` is set to generate concave polygons, there is only one `Shape2D` generated, so the start index and the end index are the same.

- **int get_collision_object_last_shape ( ) const**
  
  Return the index of the last shape generated by the editor.

- **Vector2Array get_polygon ( ) const**
  
  Return the list of points that define the polygon.

- **bool is_trigger ( ) const**
  
  Return whether this polygon is a trigger.

- **void set_build_mode ( int build_mode )**
  
  Set whether the polygon is to be a `ConvexPolygonShape2D` (build_mode==0), or a `ConcavePolygonShape2D` (build_mode==1).

- **void set_polygon ( Vector2Array polygon )**
  
  Set the array of points forming the polygon.

  When editing the point list via the editor, depending on `get_build_mode`, it has to be a list of points (for build_mode==0), or a list of lines (for build_mode==1). In the second case, the even elements of the array define the start point of the line, and the odd elements the end point.
Set whether this polygon is a trigger. A trigger polygon detects collisions, but is otherwise unaffected by physics (i.e. colliding objects will not get blocked).

12.55 CollisionShape

Inherits: Spatial < Node < Object
Category: Core

12.55.1 Brief Description

12.55.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>get_collision_object_shape_index() const</td>
</tr>
<tr>
<td>Object</td>
<td>get_shape() const</td>
</tr>
<tr>
<td>bool</td>
<td>is_trigger() const</td>
</tr>
<tr>
<td>void</td>
<td>make_convex_from_brothers()</td>
</tr>
<tr>
<td>void</td>
<td>resource_changed(Object resource)</td>
</tr>
<tr>
<td>void</td>
<td>set_shape(Object shape)</td>
</tr>
<tr>
<td>void</td>
<td>set_trigger(bool enable)</td>
</tr>
</tbody>
</table>

12.55.3 Member Function Description

- int get_collision_object_shape_index() const
- Object get_shape() const
- bool is_trigger() const
- void make_convex_from_brothers()
- void resource_changed(Object resource)
- void set_shape(Object shape)
- void set_trigger(bool enable)

12.56 CollisionShape2D

Inherits: Node2D < CanvasItem < Node < Object
Category: Core

12.56.1 Brief Description

Editor-only class for easy editing of shapes.
12.56.2 Member Functions

<table>
<thead>
<tr>
<th></th>
<th>Member Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>get_collision_object_shape_index() const</td>
</tr>
<tr>
<td>Object</td>
<td>get_shape() const</td>
</tr>
<tr>
<td>bool</td>
<td>is_trigger() const</td>
</tr>
<tr>
<td>void</td>
<td>set_shape(Object shape)</td>
</tr>
<tr>
<td>void</td>
<td>set_trigger(bool enable)</td>
</tr>
</tbody>
</table>

12.56.3 Description

Editor-only class. This is not present when running the game. It’s used in the editor to properly edit and position collision shapes in *CollisionObject2D*. This is not accessible from regular code.

12.56.4 Member Function Description

- **int get_collision_object_shape_index() const**
  
  Return the index of this shape inside its container *CollisionObject2D*. This can be used to directly access the underlying *Shape2D*.

- **Object get_shape() const**
  
  Return this shape’s *Shape2D*.

- **bool is_trigger() const**
  
  Return whether this shape is a trigger.

- **void set_shape(Object shape)**
  
  Set this shape’s *Shape2D*. This will not appear as a node, but can be directly edited as a property.

- **void set_trigger(bool enable)**
  
  Set whether this shape is a trigger. A trigger shape detects collisions, but is otherwise unaffected by physics (i.e. will not block movement of colliding objects).

12.57 Color

Category: Built-In Types

12.57.1 Brief Description

Color in RGBA format.
12.57.2 Member Functions

<table>
<thead>
<tr>
<th>Member Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color Color ( float r, float g, float b, float a )</td>
<td>Construct the color from an RGBA profile.</td>
</tr>
<tr>
<td>Color Color ( float r, float g, float b )</td>
<td>Construct the color from an RGBA profile.</td>
</tr>
<tr>
<td>Color Color ( int from )</td>
<td>Construct the color from an RGBA profile.</td>
</tr>
<tr>
<td>Color blend ( Color over )</td>
<td></td>
</tr>
<tr>
<td>float gray ( )</td>
<td></td>
</tr>
<tr>
<td>Color inverted ( )</td>
<td></td>
</tr>
<tr>
<td>Color linear_interpolate ( Color b, float t )</td>
<td></td>
</tr>
<tr>
<td>int to_32 ( )</td>
<td></td>
</tr>
<tr>
<td>int to_ARGB32 ( )</td>
<td></td>
</tr>
<tr>
<td>String to_html ( bool with_alpha=True )</td>
<td></td>
</tr>
</tbody>
</table>

12.57.3 Member Variables

- float a - Alpha (0 to 1)
- int a8 - Alpha (0 to 255)
- float b - Blue (0 to 1)
- int b8 - Blue (0 to 255)
- float g - Green (0 to 1)
- int g8 - Green (0 to 255)
- float h - Hue (0 to 1)
- float r - Red (0 to 1)
- int r8 - Red (0 to 255)
- float s - Saturation (0 to 1)
- float v - Value (0 to 1)

12.57.4 Description

A color is represented as red, green and blue (r,g,b) components. Additionally, “a” represents the alpha component, often used for transparency. Values are in floating point and usually range from 0 to 1. Some methods (such as set_modulate() ) may accept values > 1.

12.57.5 Member Function Description

- Color Color ( float r, float g, float b, float a )
  Construct the color from an RGBA profile.

- Color Color ( float r, float g, float b )
  Construct the color from an RGBA profile.

- Color Color ( int from )
  Construct the color from an RGBA profile.
• **Color Color (String from)**
Construct the color from an RGBA profile.

• **Color blend (Color over)**
Return a new color blended with another one.

• **Color contrasted ()**
Return the most contrasting color with this one.

• **float gray ()**
Convert the color to gray.

• **Color inverted ()**
Return the inverted color (1-r, 1-g, 1-b, 1-a).

• **Color linear_interpolate (Color b, float t)**
Return the linear interpolation with another color.

• **int to_32 ()**
Convert the color to a 32 bits integer (each byte represents a RGBA).

• **int to_ARGB32 ()**
Convert color to ARGB32, more compatible with DirectX.

• **String to_html (bool with_alpha=True)**
Return the HTML hexadecimal color string.

### 12.58 ColorArray

**Category:** Built-In Types

#### 12.58.1 Brief Description

Array of Colors

#### 12.58.2 Member Functions

<table>
<thead>
<tr>
<th>ColorArray</th>
<th>ColorArray (Array from)</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>append (Color color)</td>
</tr>
<tr>
<td>void</td>
<td>append_array (ColorArray array)</td>
</tr>
<tr>
<td>int</td>
<td>insert (int idx, Color color)</td>
</tr>
<tr>
<td>void</td>
<td>invert ()</td>
</tr>
<tr>
<td>void</td>
<td>push_back (Color color)</td>
</tr>
<tr>
<td>void</td>
<td>remove (int idx)</td>
</tr>
<tr>
<td>void</td>
<td>resize (int idx)</td>
</tr>
<tr>
<td>void</td>
<td>set (int idx, Color color)</td>
</tr>
<tr>
<td>int</td>
<td>size ()</td>
</tr>
</tbody>
</table>
12.58.3 Description

Array of Color, can only contains colors. Optimized for memory usage, can’t fragment the memory.

12.58.4 Member Function Description

- **ColorArray** `ColorArray (Array from)`
  Create from a generic array.
  - void `append (Color color)`
    Append an element at the end of the array (alias of `push_back`).
  - void `append_array (ColorArray array)`
    Append an `ColorArray` at the end of this array.
  - int `insert (int idx, Color color)`
    Insert a new element at a given position in the array. The position must be valid, or at the end of the array (pos==size()).
  - void `invert ()`
    Reverse the order of the elements in the array (so first element will now be the last).
  - void `push_back (Color color)`
    Append a value to the array.
  - void `remove (int idx)`
    Remove an element from the array by index.
  - void `resize (int idx)`
    Set the size of the `ColorArray`. If larger than the current size it will reserve some space beforehand, and if it is smaller it will cut off the array.
  - void `set (int idx, Color color)`
    Change the `Color` at the given index.
  - int `size ()`
    Return the array size.

12.59 ColorFrame

**Inherits:** `Control < CanvasItem < Node < Object`

**Category:** Core

12.59.1 Brief Description

12.59.2 Member Functions

<table>
<thead>
<tr>
<th>Color</th>
<th>get_frame_color () const</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>set_frame_color (Color color)</td>
</tr>
</tbody>
</table>
12.59.3 Member Function Description

- `Color get_frame_color() const`
- `void set_frame_color(Color color)`

12.60 ColorPicker

*Inherits:* `BoxContainer` `<` `Container` `<` `Control` `<` `CanvasItem` `<` `Node` `<` `Object`  

*Category:* Core

12.60.1 Brief Description

Color picker control.

12.60.2 Member Functions

<table>
<thead>
<tr>
<th>void</th>
<th>add_preset(Color arg0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color</td>
<td>get_color() const</td>
</tr>
<tr>
<td>bool</td>
<td>is_editing_alpha() const</td>
</tr>
<tr>
<td>bool</td>
<td>is_raw_mode() const</td>
</tr>
<tr>
<td>void</td>
<td>set_color(Color color)</td>
</tr>
<tr>
<td>void</td>
<td>set_edit_alpha(bool show)</td>
</tr>
<tr>
<td>void</td>
<td>set_raw_mode(bool mode)</td>
</tr>
</tbody>
</table>

12.60.3 Signals

- `color_changed(Color color)`

  Emitted when the color is changed.

12.60.4 Description

This is a simple color picker *Control*. It’s useful for selecting a color from an RGB/RGBA colorspace.

12.60.5 Member Function Description

- `void add_preset(Color arg0)`

  Adds the current selected to color to a list of colors (presets), the presets will be displayed in the color picker and the user will be able to select them, notice that the presets list is only for this color picker.

- `Color get_color() const`

  Return the current (edited) color.

- `bool is_editing_alpha() const`

  Returns whether the color has transparency or not.
Returns whether this color picker is in raw mode or not, raw mode will allow the color R, G, B component values to go beyond 1, you have to consider that the max value for color components is 1, going beyond that value will not have effect in the color, but can be used for special operations that require it (like tinting without darkening or rendering sprites in HDR).

- **void set_color ( Color color )**
  Select the current color.

- **void set_edit_alpha ( bool show )**
  Set true if you want the color to have an alpha channel (transparency), or false if you want a solid color.

- **void set_raw_mode ( bool mode )**
  Set whether this color picker is using raw mode or not, see is_raw_mode.

### 12.61 ColorPickerButton

**Inherits:** Button < BaseButton < Control < CanvasItem < Node < Object

**Category:** Core

#### 12.61.1 Brief Description

Button that pops out a ColorPicker

#### 12.61.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color</td>
<td>get_color ( ) const</td>
</tr>
<tr>
<td>ColorPicker</td>
<td>get_picker ( )</td>
</tr>
<tr>
<td>bool</td>
<td>is_editing_alpha ( ) const</td>
</tr>
<tr>
<td>void</td>
<td>set_color ( Color color )</td>
</tr>
<tr>
<td>void</td>
<td>set_edit_alpha ( bool show )</td>
</tr>
</tbody>
</table>

#### 12.61.3 Signals

- **color_changed ( Color color )**
  Emitted when the color is changed.

#### 12.61.4 Description

Encapsulates a ColorPicker making it accesible by pressing a button, pressing the button will toggle the ColorPicker visibility

#### 12.61.5 Member Function Description

- **Color get_color ( ) const**
  Gets the current color
  
- **ColorPicker get_picker ( )**
• `bool is_editing_alpha()` const

See ColorPicker.is_edit_alpha

• void `set_color(Color color)`

Sets the current color

• void `set_edit_alpha(bool show)`

See ColorPicker.set_edit_alpha

### 12.62 ColorRamp

**Inherits:** Resource < Reference < Object

**Category:** Core

#### 12.62.1 Brief Description

Color interpolator node

#### 12.62.2 Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Color get_color(int point)</code> const</td>
<td>Returns the color of the ramp color at index point</td>
</tr>
<tr>
<td><code>ColorArray get_colors()</code> const</td>
<td></td>
</tr>
<tr>
<td><code>float get_offset(int point)</code> const</td>
<td></td>
</tr>
<tr>
<td><code>RealArray get_offsets()</code> const</td>
<td></td>
</tr>
<tr>
<td><code>int get_point_count()</code> const</td>
<td></td>
</tr>
<tr>
<td><code>Color interpolate(float offset)</code></td>
<td></td>
</tr>
<tr>
<td><code>void remove_point(int offset)</code></td>
<td></td>
</tr>
<tr>
<td><code>void set_color(int point, Color color)</code></td>
<td>Adds the specified color to the end of the ramp, with the specified offset</td>
</tr>
<tr>
<td><code>void set_colors(ColorArray colors)</code></td>
<td></td>
</tr>
<tr>
<td><code>void set_offset(int point, float offset)</code></td>
<td></td>
</tr>
<tr>
<td><code>void set_offsets(RealArray offsets)</code></td>
<td></td>
</tr>
</tbody>
</table>

#### 12.62.3 Description

Given a set of colors, this node will interpolate them in order, meaning, that if you have color 1, color 2 and color 3, the ramp will interpolate (generate the colors between two colors) from color 1 to color 2 and from color 2 to color 3. Initially the ramp will have 2 colors (black and white), one (black) at ramp lower offset offset 0 and the other (white) at the ramp higher offset 1.

#### 12.62.4 Member Function Description

• void `add_point(float offset, Color color)`

Adds the specified color to the end of the ramp, with the specified offset

• `Color get_color(int point)` const

Returns the color of the ramp color at index point
• **ColorArray get_colors ( ) const**
  Returns the colors in the ramp

• **float get_offset ( int point ) const**
  Returns the offset of the ramp color at index `point`

• **RealArray get_offsets ( ) const**
  Returns the offsets for the colors in this ramp

• **int get_point_count ( ) const**
  Returns the number of colors in the ramp

• **Color interpolate ( float offset )**
  Returns the interpolated color specified by `offset`

• **void remove_point ( int offset )**
  Removes the color at the index `offset`

• **void set_color ( int point, Color color )**
  Sets the color of the ramp color at index `point`

• **void set_colors ( ColorArray colors )**
  Sets the colors for the specified amount of elements. Calling this function with a different number of elements than previously defined causes the ramp to resize its colors and offsets array to accomodate the new elements.

• **void set_offset ( int point, float offset )**
  Sets the offset for the ramp color at index `point`

• **void set_offsets ( RealArray offsets )**
  Sets the offset for the specified amount of elements. Calling this function with a different number of elements than previously defined causes the ramp to resize its colors and offsets array to accomodate the new elements, all new colors will be black by default.

### 12.63 ConcavePolygonShape

**Inherits:** Shape < Resource < Reference < Object

**Category:** Core

#### 12.63.1 Brief Description

Concave polygon shape.

#### 12.63.2 Member Functions

<table>
<thead>
<tr>
<th>Vector3Array</th>
<th>get_faces ( ) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>set_faces ( Vector3Array faces )</td>
</tr>
</tbody>
</table>
12.63.3 Description

Concave polygon shape resource, which can be set into a PhysicsBody or area. This shape is created by feeding a list of triangles.

12.63.4 Member Function Description

- **Vector3Array get_faces() const**
  
  Return the faces (an array of triangles).

- **void set_faces(Vector3Array faces)**
  
  Set the faces (an array of triangles).

12.64 ConcavePolygonShape2D

**Inherits:** Shape2D < Resource < Reference < Object

**Category:** Core

12.64.1 Brief Description

Concave polygon 2D shape resource for physics.

12.64.2 Member Functions

<table>
<thead>
<tr>
<th>Vector2Array</th>
<th>get_segments() const</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>set_segments(Vector2Array segments)</td>
</tr>
</tbody>
</table>

12.64.3 Description

Concave polygon 2D shape resource for physics. It is made out of segments and is very optimal for complex polygonal concave collisions. It is really not advised to use for RigidBody2D nodes. A CollisionPolygon2D in convex decomposition mode (solids) or several convex objects are advised for that instead. Otherwise, a concave polygon 2D shape is better for static collisions.

The main difference between a ConvexPolygonShape2D and a ConcavePolygonShape2D is that a concave polygon assumes it is concave and uses a more complex method of collision detection, and a convex one forces itself to be convex in order to speed up collision detection.

12.64.4 Member Function Description

- **Vector2Array get_segments() const**
  
  Return the array of segments.

- **void set_segments(Vector2Array segments)**
  
  Set the array of segments.
12.65 ConeTwistJoint

Inherits: Joint < Spatial < Node < Object
Category: Core

12.65.1 Brief Description

12.65.2 Member Functions

| float | get_param ( int param ) const |
| void  | set_param ( int param, float value ) |

12.65.3 Numeric Constants

- PARAM_SWING_SPAN = 0
- PARAM_TWIST_SPAN = 1
- PARAM_BIAS = 2
- PARAM_SOFTNESS = 3
- PARAM_RELAXATION = 4
- PARAM_MAX = 5

12.65.4 Member Function Description

- float get_param ( int param ) const
- void set_param ( int param, float value )

12.66 ConfigFile

Inherits: Reference < Object
Category: Core

12.66.1 Brief Description

Helper class to handle INI-style files.
12.66.2 Member Functions

<table>
<thead>
<tr>
<th>StringArray</th>
<th>get_section_keys ( String section ) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>StringArray</td>
<td>get_sections() const</td>
</tr>
<tr>
<td>Variant</td>
<td>get_value ( String section, String key, Variant default=NULL ) const</td>
</tr>
<tr>
<td>bool</td>
<td>has_section ( String section ) const</td>
</tr>
<tr>
<td>bool</td>
<td>has_section_key ( String section, String key ) const</td>
</tr>
<tr>
<td>Error</td>
<td>load ( String path )</td>
</tr>
<tr>
<td>Error</td>
<td>save ( String path )</td>
</tr>
<tr>
<td>void</td>
<td>set_value ( String section, String key, Variant value )</td>
</tr>
</tbody>
</table>

12.66.3 Description

This helper class can be used to store Variant values on the filesystem using an INI-style formatting. The stored values as referenced by a section and a key. The stored data can be saved to or parsed from a file, though ConfigFile objects can also be used directly with accessing the filesystem.

The following example shows how to parse a INI-style file from the system, read its contents and store new values in it:

```plaintext
var config = ConfigFile.new()
var err = config.load("user://settings.cfg")
if err == OK:
    # if not, something went wrong with the file loading
    var screen_width = get_value("display", "width", 1024)
    # Look for the display/width pair, and default to 1024 if missing
    var config.has_section_key("audio", "mute"):
        config.set_value("audio", "mute", false)
    # Save the changes by overwriting the previous file
    config.save("user://settings.cfg")
```

12.66.4 Member Function Description

- **StringArray get_section_keys ( String section ) const**
  Return an array of all defined key identifiers in the specified section.

- **StringArray get_sections() const**
  Return an array of all defined section identifiers.

- **Variant get_value ( String section, String key, Variant default=NULL ) const**
  Return the current value for the specified section and key. If the section and/or the key do not exist, the method returns the value of the optional default argument (and thus NULL if not specified).

- **bool has_section ( String section ) const**
  Check if the specified section exists.

- **bool has_section_key ( String section, String key ) const**
  Check if the specified section-key pair exists.

- **Error load ( String path )**
Load the config file specified as a parameter. The file’s contents are parsed and loaded in the ConfigFile object from which the method was called. The return value is one of the OK, FAILED or ERR_* constants listed in @Global Scope (if the load was successful, it returns OK).

- Error `save ( String path )`

Save the contents of the ConfigFile object to the file specified as a parameter. The output file uses an INI-style structure. The return value is one of the OK, FAILED or ERR_* constants listed in @Global Scope (if the save was successful, it returns OK).

- void `set_value ( String section, String key, Variant value )`

Assign a value to the specified key of the the specified section. If the section and/or the key do not exist, they are created. Passing a NULL value deletes the specified key if it exists (and deletes the section if it ends up empty once the key has been removed).

12.67 ConfirmationDialog

Inherits: `AcceptDialog < WindowDialog < Popup < Control < CanvasItem < Node < Object`

Inherited By: `EditorFileDialog, FileDialog`

Category: Core

12.67.1 Brief Description

Dialog for confirmation of actions.

12.67.2 Member Functions

```
Button get_cancel ()
```

12.67.3 Description

Dialog for confirmation of actions. This dialog inherits from `AcceptDialog`, but has by default an OK and Cancel button (in host OS order).

12.67.4 Member Function Description

- `Button get_cancel ()`

Return the cancel button.

12.68 Container

Inherits: `Control < CanvasItem < Node < Object`

Inherited By: `PanelContainer, GridContainer, ScrollContainer, MarginContainer, CenterContainer, GraphNode, SplitContainer, BoxContainer`

Category: Core
12.68.1 Brief Description

Base node for containers.

12.68.2 Member Functions

```cpp
void fit_child_in_rect (Control child, Rect2 rect)
void queue_sort ()
```

12.68.3 Signals

- `sort_children ()`

Emitted when sorting the children is needed.

12.68.4 Numeric Constants

- `NOTIFICATION_SORT_CHILDREN = 50` — Notification for when sorting the children, it must be obeyed immediately.

12.68.5 Description

Base node for containers. A `Container` contains other controls and automatically arranges them in a certain way.

A Control can inherit this to create custom container classes.

12.68.6 Member Function Description

- `void fit_child_in_rect (Control child, Rect2 rect)`

Fit a child control in a given rect. This is mainly a helper for creating custom container classes.

- `void queue_sort ()`

Queue resort of the contained children. This is called automatically anyway, but can be called upon request.

12.69 Control

**Inherits:** `CanvasItem < Node < Object`

**Inherited By:** `Label, Tabs, TextureFrame, ButtonArray, VideoPlayer, LineEdit, ColorFrame, Container, ReferenceFrame, Patch9Frame, TextEdit, BaseButton, Popup, Tree, Separator, Panel, TabContainer, Range, RichTextLabel, GraphEdit, ItemList`

**Category:** Core

12.69.1 Brief Description

Control is the base node for all the GUI components.
## 12.69.2 Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Declaration</th>
</tr>
</thead>
<tbody>
<tr>
<td>void <strong>_input_event</strong> (InputEvent event)</td>
<td>virtual</td>
</tr>
<tr>
<td>void <strong>accept_event</strong> ()</td>
<td></td>
</tr>
<tr>
<td>void <strong>add_color_override</strong> (String name, Color color)</td>
<td></td>
</tr>
<tr>
<td>void <strong>add_constant_override</strong> (String name, int constant)</td>
<td></td>
</tr>
<tr>
<td>void <strong>add_font_override</strong> (String name, Font font)</td>
<td></td>
</tr>
<tr>
<td>void <strong>add_icon_override</strong> (String name, Texture texture)</td>
<td></td>
</tr>
<tr>
<td>void <strong>add_shader_override</strong> (String name, Shader shader)</td>
<td></td>
</tr>
<tr>
<td>void <strong>add_style_override</strong> (String name, StyleBox stylebox)</td>
<td></td>
</tr>
<tr>
<td>bool <strong>can_drop_data</strong> (Vector2 pos, Variant data)</td>
<td>virtual</td>
</tr>
<tr>
<td>void <strong>drop_data</strong> (Vector2 pos, Variant data)</td>
<td>virtual</td>
</tr>
<tr>
<td>void <strong>force_drag</strong> (Variant data, Object preview)</td>
<td></td>
</tr>
<tr>
<td>int <strong>get_anchor</strong> (int margin) const</td>
<td></td>
</tr>
<tr>
<td>Vector2 <strong>get_begin</strong> () const</td>
<td></td>
</tr>
<tr>
<td>Color <strong>get_color</strong> (String name, String type=&quot;&quot;) const</td>
<td></td>
</tr>
<tr>
<td>Vector2 <strong>get_combined_minimum_size</strong> () const</td>
<td></td>
</tr>
<tr>
<td>int <strong>get_constant</strong> (String name, String type=&quot;&quot;) const</td>
<td></td>
</tr>
<tr>
<td>int <strong>get_cursor_shape</strong> (Vector2 pos=Vector2(0, 0) const</td>
<td></td>
</tr>
<tr>
<td>Vector2 <strong>get_custom_minimum_size</strong> () const</td>
<td></td>
</tr>
<tr>
<td>int <strong>get_default_cursor_shape</strong> () const</td>
<td></td>
</tr>
<tr>
<td>Object <strong>get_drag_data</strong> (Vector2 pos) virtual</td>
<td></td>
</tr>
<tr>
<td>Vector2 <strong>get_end</strong> () const</td>
<td></td>
</tr>
<tr>
<td>int <strong>get_focus_mode</strong> () const</td>
<td></td>
</tr>
<tr>
<td>NodePath <strong>get_focus_neighbour</strong> (int margin) const</td>
<td></td>
</tr>
<tr>
<td>Control <strong>get_focus_owner</strong> () const</td>
<td></td>
</tr>
<tr>
<td>Font <strong>get_font</strong> (String name, String type=&quot;&quot;) const</td>
<td></td>
</tr>
<tr>
<td>Vector2 <strong>get_global_pos</strong> () const</td>
<td></td>
</tr>
<tr>
<td>Rect2 <strong>get_global_rect</strong> () const</td>
<td></td>
</tr>
<tr>
<td>int <strong>get_h_size_flags</strong> () const</td>
<td></td>
</tr>
<tr>
<td>Texture <strong>get_icon</strong> (String name, String type=&quot;&quot;) const</td>
<td></td>
</tr>
<tr>
<td>float <strong>get_margin</strong> (int margin) const</td>
<td></td>
</tr>
<tr>
<td>Vector2 <strong>get_minimum_size</strong> () const</td>
<td></td>
</tr>
<tr>
<td>Vector2 <strong>get_minimum_size</strong> () virtual</td>
<td></td>
</tr>
<tr>
<td>Vector2 <strong>get_parent_area_size</strong> () const</td>
<td></td>
</tr>
<tr>
<td>Control <strong>get_parent_control</strong> () const</td>
<td></td>
</tr>
<tr>
<td>Vector2 <strong>get_pos</strong> () const</td>
<td></td>
</tr>
<tr>
<td>Rect2 <strong>get_rect</strong> () const</td>
<td></td>
</tr>
<tr>
<td>float <strong>get_rotation</strong> () const</td>
<td></td>
</tr>
<tr>
<td>float <strong>get_rotation_deg</strong> () const</td>
<td></td>
</tr>
<tr>
<td>Vector2 <strong>get_scale</strong> () const</td>
<td></td>
</tr>
<tr>
<td>Vector2 <strong>get_size</strong> () const</td>
<td></td>
</tr>
<tr>
<td>float <strong>get_stretch_ratio</strong> () const</td>
<td></td>
</tr>
<tr>
<td>StyleBox <strong>get_stylebox</strong> (String name, String type=&quot;&quot;) const</td>
<td></td>
</tr>
<tr>
<td>Theme <strong>get_theme</strong> () const</td>
<td></td>
</tr>
<tr>
<td>String <strong>get_tooltip</strong> (Vector2 atpos=Vector2(0, 0) const</td>
<td></td>
</tr>
<tr>
<td>int <strong>get_v_size_flags</strong> () const</td>
<td></td>
</tr>
<tr>
<td>void <strong>grab_click_focus</strong> ()</td>
<td></td>
</tr>
<tr>
<td>void <strong>grab_focus</strong> ()</td>
<td></td>
</tr>
<tr>
<td>bool <strong>has_color</strong> (String name, String type=&quot;&quot;) const</td>
<td></td>
</tr>
<tr>
<td>bool <strong>has_color_override</strong> (String name) const</td>
<td></td>
</tr>
</tbody>
</table>

Continued on next page
### Table 12.12 – continued from previous page

| bool | has_constant (String name, String type="") const |
| bool | has_constant_override (String name) const |
| bool | has_focus () const |
| bool | has_font (String name, String type="") const |
| bool | has_font_override (String name) const |
| bool | has_icon (String name, String type="") const |
| bool | has_icon_override (String name) const |
| void | has_point (Vector2 point) virtual |
| bool | has_stylebox (String name, String type="") const |
| bool | has_stylebox_override (String name) const |
| bool | is_ignoring_mouse () const |
| bool | is_stopping_mouse () const |
| void | minimum_size_changed () |
| void | release_focus () |
| void | set_anchor (int margin, int anchor_mode, bool keep_margin=false) |
| void | set_anchor_and_margin (int margin, int anchor_mode, float offset) |
| void | set_area_as_parent_rect (int margin=0) |
| void | set_begin (Vector2 pos) |
| void | set_custom_minimum_size (Vector2 size) |
| void | set_default_cursor_shape (int shape) |
| void | set_drag_forwarding (Control target) |
| void | set_drag_preview (Control control) |
| void | set_end (Vector2 pos) |
| void | set_focus_mode (int mode) |
| void | set_focus_neighbour (int margin, NodePath neighbour) |
| void | set_global_pos (Vector2 pos) |
| void | set_h_size_flags (int flags) |
| void | set_ignore_mouse (bool ignore) |
| void | set_margin (int margin, float offset) |
| void | set_pos (Vector2 pos) |
| void | set_rotation (float radians) |
| void | set_rotation_deg (float degrees) |
| void | set_scale (Vector2 scale) |
| void | set_size (Vector2 size) |
| void | set_stop_mouse (bool stop) |
| void | set_stretch_ratio (float ratio) |
| void | set_theme (Theme theme) |
| void | set_tooltip (String tooltip) |
| void | set_v_size_flags (int flags) |
| void | show_modal (bool exclusive=false) |
| void | warp_mouse (Vector2 to_pos) |

### 12.69.3 Signals

- **focus_enter ()**
  
  Emitted when keyboard focus is gained.

- **focus_exit ()**
  
  Emitted when the keyboard focus is lost.

- **input_event (InputEvent ev)**
Emitted when an input event is received. Connecting in realtime is recommended for accepting the events.

- **minimum_size_changed()**

Emitted when the minimum size of the control changed.

- **modal_close()**
- **mouse_enter()**

Emitted when the mouse enters the control area.

- **mouse_exit()**

Emitted when the mouse left the control area.

- **resized()**

Emitted when the control changed size.

- **size_flags_changed()**

Emitted when the size flags changed.

### 12.69.4 Numeric Constants

- **ANCHOR_BEGIN** = 0 — X is relative to MARGIN_LEFT, Y is relative to MARGIN_TOP.
- **ANCHOR_END** = 1 — X is relative to -MARGIN_RIGHT, Y is relative to -MARGIN_BOTTOM.
- **ANCHOR_RATIO** = 2 — X and Y are a ratio (0 to 1) relative to the parent size 0 is left/top, 1 is right/bottom.
- **ANCHOR_CENTER** = 3
- **FOCUS_NONE** = 0 — Control can’t acquire focus.
- **FOCUS_CLICK** = 1 — Control can acquire focus only if clicked.
- **FOCUS_ALL** = 2 — Control can acquire focus if clicked, or by pressing TAB/Directionals in the keyboard from another Control.
- **NOTIFICATION_RESIZED** = 40 — Control changed size (get_size() reports the new size).
- **NOTIFICATION_MOUSE_ENTER** = 41 — Mouse pointer entered the area of the Control.
- **NOTIFICATION_MOUSE_EXIT** = 42 — Mouse pointer exited the area of the Control.
- **NOTIFICATION_FOCUS_ENTER** = 43 — Control gained focus.
- **NOTIFICATION_FOCUS_EXIT** = 44 — Control lost focus.
- **NOTIFICATION_THEME_CHANGED** = 45 — Theme changed. Redrawing is desired.
- **NOTIFICATION_MODAL_CLOSE** = 46 — Modal control was closed.
- **CURSOR_ARROW** = 0
- **CURSOR_IBEAM** = 1
- **CURSOR_POINTING_HAND** = 2
- **CURSOR_CROSS** = 3
- **CURSOR_WAIT** = 4
- **CURSOR_BUSY** = 5
- **CURSOR_DRAG** = 6
12.69.5 Description

Control is the base class Node for all the GUI components. Every GUI component inherits from it, directly or indirectly. In this way, sections of the scene tree made of contiguous control nodes, become user interfaces.

Controls are relative to the parent position and size by using anchors and margins. This ensures that they can adapt easily in most situation to changing dialog and screen sizes. When more flexibility is desired, Container derived nodes can be used.

Anchors work by defining which margin do they follow, and a value relative to it. Allowed anchoring modes are ANCHOR_BEGIN, where the margin is relative to the top or left margins of the parent (in pixels), ANCHOR_END for the right and bottom margins of the parent and ANCHOR_RATIO, which is a ratio from 0 to 1 in the parent range.

Input device events (InputEvent) are first sent to the root controls via the Node._input, which distribute it through the tree, then delivers them to the adequate one (under cursor or keyboard focus based) by calling MainLoop._input_event. There is no need to enable input processing on controls to receive such events. To ensure that no one else will receive the event (not even Node._unhandled_input), the control can accept it by calling accept_event.

Only one control can hold the keyboard focus (receiving keyboard events), for that the control must define the focus mode with set_focus_mode. Focus is lost when another control gains it, or the current focus owner is hidden.

It is sometimes desired for a control to ignore mouse/pointer events. This is often the case when placing other controls on top of a button, in such cases. Calling set_ignore_mouse enables this function.

Finally, controls are skinned according to a Theme. Setting a Theme on a control will propagate all the skinning down the tree. Optionally, skinning can be overridden per each control by calling the add_*.override functions, or from the editor.

12.69.6 Member Function Description

- void _input_event (InputEvent event) virtual

Called when an input event reaches the control.

- void accept_event ()
Handles the event, no other control will receive it and it will not be sent to nodes waiting on `Node.unhandled_input` or `Node.unhandled_key_input`.

- `void add_color_override ( String name, Color color )`
- `void add_constant_override ( String name, int constant )`

Override a single constant (integer) in the theme of this Control. If constant equals Theme.INVALID_CONSTANT, override is cleared.

- `void add_font_override ( String name, Font font )`

Override a single font (Font) in the theme of this Control. If font is empty, override is cleared.

- `void add_icon_override ( String name, Texture texture )`

Override a single icon (Texture) in the theme of this Control. If texture is empty, override is cleared.

- `void add_shader_override ( String name, Shader shader )`
- `void add_style_override ( String name, StyleBox stylebox )`

Override a single stylebox (Stylebox) in the theme of this Control. If stylebox is empty, override is cleared.

- `bool can_drop_data ( Vector2 pos, Variant data ) virtual`
- `void drop_data ( Vector2 pos, Variant data ) virtual`
- `void force_drag ( Variant data, Object preview )`
- `int get_anchor ( int margin ) const`

Return the anchor type (ANCHOR_BEGIN, ANCHOR_END, ANCHOR_RATIO) for a given margin (MARGIN_LEFT, MARGIN_TOP, MARGIN_RIGHT, MARGIN_BOTTOM).

- `Vector2 get_begin ( ) const`
- `Color get_color ( String name, String type="" ) const`
- `Vector2 get_combined_minimum_size ( ) const`
- `int get_constant ( String name, String type="" ) const`
- `int get_cursor_shape ( Vector2 pos=Vector2(0, 0) ) const`

Return the cursor shape at a certain position in the control.

- `Vector2 get_custom_minimum_size ( ) const`
- `int get_default_cursor_shape ( ) const`

Return the default cursor shape for this control. See enum CURSOR_* for the list of shapes.

- `Object get_drag_data ( Vector2 pos ) virtual`
- `Vector2 get_end ( ) const`

Returns MARGIN_LEFT and MARGIN_TOP at the same time. This is a helper (see `set_margin`).

- `int get_focus_mode ( ) const`

Returns the focus access mode for the control (FOCUS_NONE, FOCUS_CLICK, FOCUS_ALL) (see `set_focus_mode`).

- `NodePath get_focus_neighbour ( int margin ) const`

Return the forced neighbour for moving the input focus to. When pressing TAB or directional/joystick directions focus is moved to the next control in that direction. However, the neighbour to move to can be forced with this function.

- `Control get_focus_owner ( ) const`
Return which control is owning the keyboard focus, or null if no one.

- **Font get_font ( String name, String type="" ) const**

- **Vector2 get_global_pos ( ) const**

Returns the Control position, relative to the top-left corner of the parent Control and independent of the anchor mode.

- **Rect2 get_global_rect ( ) const**

Return position and size of the Control, relative to the top-left corner of the `window` Control. This is a helper (see `get_global_pos, get_size`).

- **int get_h_size_flags ( ) const**

Hint for containers, return horizontal positioning flags.

- **Texture get_icon ( String name, String type="" ) const**

- **float get_margin ( int margin ) const**

Return a margin offset. Margin can be one of (MARGIN_LEFT, MARGIN_TOP, MARGIN_RIGHT, MARGIN_BOTTOM). Offset value being returned depends on the anchor mode.

- **Vector2 get_minimum_size ( ) const**

Return the minimum size this Control can shrink to. A control will never be displayed or resized smaller than its minimum size.

- **Vector2 get_minimum_size ( ) virtual**

Return the minimum size this Control can shrink to. A control will never be displayed or resized smaller than its minimum size.

- **Vector2 get_parent_area_size ( ) const**

- **Control get_parent_control ( ) const**

- **Vector2 get_pos ( ) const**

Returns the Control position, relative to the top-left corner of the parent Control and independent of the anchor mode.

- **Rect2 get_rect ( ) const**

Return position and size of the Control, relative to the top-left corner of the parent Control. This is a helper (see `get_pos, get_size`).

- **float get_rotation ( ) const**

- **float get_rotation_deg ( ) const**

- **Vector2 get_scale ( ) const**

- **Vector2 get_size ( ) const**

Returns the size of the Control, computed from all margins, however the size returned will **never be smaller than the minimum size reported by** `get_minimum_size`. This means that even if end position of the Control rectangle is smaller than the begin position, the Control will still display and interact correctly. (see description, `get_minimum_size, set_margin, set_anchor`).

- **float get_stretch_ratio ( ) const**

Hint for containers, return the stretch ratio. This value is relative to other stretch ratio, so if this control has 2 and another has 1, this one will be twice as big.

- **StyleBox get_stylebox ( String name, String type="" ) const**

- **Theme get_theme ( ) const**
Return a Theme override, if one exists (see set_theme).

- **String get_tooltip ( Vector2 atpos=Vector2(0, 0) ) const**

Return the tooltip, which will appear when the cursor is resting over this control.

- **int get_v_size_flags ( ) const**

Hint for containers, return vertical positioning flags.

- void grab_click_focus ( )

- void grab_focus ( )

Steal the focus from another control and become the focused control (see set_focus_mode).

- **bool has_color ( String name, String type="" ) const**

- **bool has_color_override ( String name ) const**

- **bool has_constant ( String name, String type="" ) const**

- **bool has_constant_override ( String name ) const**

- **bool has_focus ( ) const**

Return whether the Control is the current focused control (see set_focus_mode).

- **bool has_font ( String name, String type="" ) const**

- **bool has_font_override ( String name ) const**

- **bool has_icon ( String name, String type="" ) const**

- **bool has_icon_override ( String name ) const**

- void has_point ( Vector2 point ) virtual

- **bool has_stylebox ( String name, String type="" ) const**

- **bool has_stylebox_override ( String name ) const**

- **bool is_ignoring_mouse ( ) const**

Return if the control is ignoring mouse events (even touchpad events send mouse events).

- **bool is_stopping_mouse ( ) const**

- void minimum_size_changed ( )

- void release_focus ( )

Give up the focus, no other control will be able to receive keyboard input.

- void set_anchor ( int margin, int anchor_mode, bool keep_margin=false )

Change the anchor (ANCHOR_BEGIN, ANCHOR_END, ANCHOR_RATIO) type for a margin (MARGIN_LEFT, MARGIN_TOP, MARGIN_RIGHT, MARGIN_BOTTOM). Changing the anchor mode converts the current margin offset from the previous anchor mode to the new one, so margin offsets (set_margin) must be done after setting anchors, or at the same time (set_anchor_and_margin)

Additionally, keep_margin controls whether margins should be left the same, or changed to keep the same position and size on-screen.

- void set_anchor_and_margin ( int margin, int anchor_mode, float offset )

Change the anchor (ANCHOR_BEGIN, ANCHOR_END, ANCHOR_RATIO) type for a margin (MARGIN_LEFT, MARGIN_TOP, MARGIN_RIGHT, MARGIN_BOTTOM), and also set its offset. This is a helper (see set_anchor and set_margin).
• void `set_area_as_parent_rect` (int margin=0)
Change all margins and anchors, so this Control always takes up the same area as the parent Control. This is a helper (see `set_anchor`, `set_margin`).

• void `set_begin` (Vector2 pos)
Sets MARGIN_LEFT and MARGIN_TOP at the same time. This is a helper (see `set_margin`).

• void `set_custom_minimum_size` (Vector2 size)
• void `set_default_cursor_shape` (int shape)
Set the default cursor shape for this control. See enum CURSOR_* for the list of shapes.

• void `set_drag_forwarding` (Control target)
• void `set_drag_preview` (Control control)
• void `set_end` (Vector2 pos)
Sets MARGIN_RIGHT and MARGIN_BOTTOM at the same time. This is a helper (see `set_margin`).

• void `set_focus_mode` (int mode)
Set the focus access mode for the control (FOCUS_NONE, FOCUS_CLICK, FOCUS_ALL). Only one Control can be focused at the same time, and it will receive keyboard signals.

• void `set_focus_neighbour` (int margin, NodePath neighbour)
Force a neighbour for moving the input focus to. When pressing TAB or directional/joypad directions focus is moved to the next control in that direction. However, the neighbour to move to can be forced with this function.

• void `set_global_pos` (Vector2 pos)
Move the Control to a new position, relative to the top-left corner of the window Control, and without changing current anchor mode. (see `set_margin`).

• void `set_h_size_flags` (int flags)
Hint for containers, set horizontal positioning flags.

• void `set_ignore_mouse` (bool ignore)
Ignore mouse events on this control (even touchpad events send mouse events).

• void `set_margin` (int margin, float offset)
Set a margin offset. Margin can be one of (MARGIN_LEFT, MARGIN_TOP, MARGIN_RIGHT, MARGIN_BOTTOM). Offset value being set depends on the anchor mode.

• void `set_pos` (Vector2 pos)
Move the Control to a new position, relative to the top-left corner of the parent Control, changing all margins if needed and without changing current anchor mode. This is a helper (see `set_margin`).

• void `set_rotation` (float radians)
• void `set_rotation_deg` (float degrees)
• void `set_scale` (Vector2 scale)
• void `set_size` (Vector2 size)
Changes MARGIN_RIGHT and MARGIN_BOTTOM to fit a given size. This is a helper (see `set_margin`).

• void `set_stop_mouse` (bool stop)
• void `set_stretch_ratio` (float ratio)
Hint for containers, set the stretch ratio. This value is relative to other stretch ratio, so if this control has 2 and another has 1, this one will be twice as big.

- void set_theme (Theme theme)
  Override whole the Theme for this Control and all its children controls.

- void set_tooltip (String tooltip)
  Set a tooltip, which will appear when the cursor is resting over this control.

- void set_v_size_flags (int flags)
  Hint for containers, set vertical positioning flags.

- void show_modal (bool exclusive=false)
  Display a Control as modal. Control must be a subwindow. Modal controls capture the input signals until closed or the area outside them is accessed. When a modal control loses focus, or the ESC key is pressed, they automatically hide. Modal controls are used extensively for popup dialogs and menus.

- void warp_mouse (Vector2 to_pos)

### 12.70 ConvexPolygonShape

**Inherits:** Shape < Resource < Reference < Object

**Category:** Core

**12.70.1 Brief Description**

Convex Polygon Shape.

**12.70.2 Member Functions**

<table>
<thead>
<tr>
<th>Vector3Array</th>
<th>get_points ( ) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>set_points ( Vector3Array points )</td>
</tr>
</tbody>
</table>

**12.70.3 Description**

Convex polygon shape resource, which can be set into a PhysicsBody or area.

**12.70.4 Member Function Description**

- Vector3Array get_points ( ) const
- void set_points ( Vector3Array points )

### 12.71 ConvexPolygonShape2D

**Inherits:** Shape2D < Resource < Reference < Object

**Category:** Core
12.71.1 Brief Description

Convex Polygon Shape for 2D physics.

12.71.2 Member Functions

<table>
<thead>
<tr>
<th>Vector2Array</th>
<th>get_points ( ) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>set_point_cloud ( Vector2Array point_cloud )</td>
</tr>
<tr>
<td>void</td>
<td>set_points ( Vector2Array points )</td>
</tr>
</tbody>
</table>

12.71.3 Description

Convex Polygon Shape for 2D physics. A convex polygon, whatever its shape, is internally decomposed into as many convex polygons as needed to ensure all collision checks against it are always done on convex polygons (which are faster to check).

The main difference between a ConvexPolygonShape2D and a ConcavePolygonShape2D is that a concave polygon assumes it is concave and uses a more complex method of collision detection, and a convex one forces itself to be convex in order to speed up collision detection.

12.71.4 Member Function Description

- Vector2Array get_points ( ) const
  Return a list of points in either clockwise or counter clockwise order, forming a convex polygon.

- void set_point_cloud ( Vector2Array point_cloud )
  Currently, this method does nothing.

- void set_points ( Vector2Array points )
  Set a list of points in either clockwise or counter clockwise order, forming a convex polygon.

12.72 CubeMap

Inherits: Resource < Reference < Object

Category: Core
### 12.72.1 Brief Description

### 12.72.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td><code>get_flags()</code> const</td>
</tr>
<tr>
<td>int</td>
<td><code>get_height()</code> const</td>
</tr>
<tr>
<td>float</td>
<td><code>get_lossy_storage_quality()</code> const</td>
</tr>
<tr>
<td>RID</td>
<td><code>get_rid()</code> const</td>
</tr>
<tr>
<td>Image</td>
<td><code>get_side(int side)</code> const</td>
</tr>
<tr>
<td>int</td>
<td><code>get_storage()</code> const</td>
</tr>
<tr>
<td>int</td>
<td><code>get_width()</code> const</td>
</tr>
<tr>
<td>void</td>
<td><code>set_flags(int flags)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_lossy_storage_quality(float quality)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_side(int side, Image image)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_storage(int mode)</code></td>
</tr>
</tbody>
</table>

### 12.72.3 Numeric Constants

- `STORAGE_RAW = 0`
- `STORAGE_COMPRESS_LOSSY = 1`
- `STORAGE_COMPRESS_LOSSLESS = 2`
- `SIDE_LEFT = 0`
- `SIDE_RIGHT = 1`
- `SIDE_BOTTOM = 2`
- `SIDE_TOP = 3`
- `SIDE_FRONT = 4`
- `SIDE_BACK = 5`
- `FLAG_MIPMAPS = 1`
- `FLAG_REPEAT = 2`
- `FLAGFILTER = 4`
- `FLAGS_DEFAULT = 7`

### 12.72.4 Member Function Description

- `int get_flags()` const
- `int get_height()` const
- `float get_lossy_storage_quality()` const
- `RID get_rid()` const
- `Image get_side(int side)` const
- `int get_storage()` const
- `int get_width()` const
- `void set_flags(int flags)`
• void set_lossy_storage_quality ( float quality )
• void set_side ( int side, Image image )
• void set_storage ( int mode )

12.73 Curve2D

Inherits: Resource < Reference < Object

Category: Core

12.73.1 Brief Description

Describes a Bezier curve in 2D space.

12.73.2 Member Functions

<table>
<thead>
<tr>
<th>void</th>
<th>add_point ( Vector2 pos, Vector2 in=Vector2(0, 0), Vector2 out=Vector2(0, 0), int atpos=-1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>clear_points ( )</td>
</tr>
<tr>
<td>float</td>
<td>get_bake_interval ( ) const</td>
</tr>
<tr>
<td>float</td>
<td>get_baked_length ( ) const</td>
</tr>
<tr>
<td>Vector2Array</td>
<td>get_baked_points ( ) const</td>
</tr>
<tr>
<td>int</td>
<td>get_point_count ( ) const</td>
</tr>
<tr>
<td>Vector2</td>
<td>get_point_in ( int idx ) const</td>
</tr>
<tr>
<td>Vector2</td>
<td>get_point_out ( int idx ) const</td>
</tr>
<tr>
<td>Vector2</td>
<td>get_point_pos ( int idx ) const</td>
</tr>
<tr>
<td>Vector2</td>
<td>interpolate ( int idx, float t ) const</td>
</tr>
<tr>
<td>Vector2</td>
<td>interpolate_baked ( float offset, bool cubic=false ) const</td>
</tr>
<tr>
<td>Vector2</td>
<td>interpolatef ( float fofs ) const</td>
</tr>
<tr>
<td>void</td>
<td>remove_point ( int idx )</td>
</tr>
<tr>
<td>void</td>
<td>set_bake_interval ( float distance )</td>
</tr>
<tr>
<td>void</td>
<td>set_point_in ( int idx, Vector2 pos )</td>
</tr>
<tr>
<td>void</td>
<td>set_point_out ( int idx, Vector2 pos )</td>
</tr>
<tr>
<td>void</td>
<td>set_point_pos ( int idx, Vector2 pos )</td>
</tr>
<tr>
<td>Vector2Array</td>
<td>tesselate ( int max_stages=5, float tolerance_degrees=4 ) const</td>
</tr>
</tbody>
</table>

12.73.3 Description

This class describes a Bezier curve in 2D space. It is mainly used to give a shape to a Path2D, but can be manually sampled for other purposes.

It keeps a cache of precalculated points along the curve, to speed further calculations up.

12.73.4 Member Function Description

• void add_point ( Vector2 pos, Vector2 in=Vector2(0, 0), Vector2 out=Vector2(0, 0), int atpos=-1 )
Adds a point to a curve, at position “pos”, with control points “in” and “out”.

If “atpos” is given, the point is inserted before the point number “atpos”, moving that point (and every point after) after the inserted point. If “atpos” is not given, or is an illegal value (atpos <0 or atpos >= get_point_count), the point will be appended at the end of the point list.

- void clear_points()
- float get_bake_interval() const

Returns the distance between two adjacent cached points.

- float get_baked_length() const

Returns the total length of the curve, based on the cached points. Given enough density (see set_bake_interval), it should be approximate enough.

- Vector2Array get_baked_points() const

Returns the cache of points as a Vector2Array.

- int get_point_count() const

Returns the number of points describing the curve.

- Vector2 get_point_in(int idx) const

Returns the position of the control point leading to the vertex “idx”. If the index is out of bounds, the function sends an error to the console, and returns (0, 0).

- Vector2 get_point_out(int idx) const

Returns the position of the control point leading out of the vertex “idx”. If the index is out of bounds, the function sends an error to the console, and returns (0, 0).

- Vector2 get_point_pos(int idx) const

Returns the position of the vertex “idx”. If the index is out of bounds, the function sends an error to the console, and returns (0, 0).

- Vector2 interpolate(int idx, float t) const

Returns the position between the vertex “idx” and the vertex “idx”+1, where “t” controls if the point is the first vertex (t = 0.0), the last vertex (t = 1.0), or in between. Values of “t” outside the range (0.0 >= t <=1) give strange, but predictable results.

If “idx” is out of bounds it is truncated to the first or last vertex, and “t” is ignored. If the curve has no points, the function sends an error to the console, and returns (0, 0).

- Vector2 interpolate_baked(float offset, bool cubic=false) const

Returns a point within the curve at position “offset”, where “offset” is measured as a pixel distance along the curve.

To do that, it finds the two cached points where the “offset” lies between, then interpolates the values. This interpolation is cubic if “cubic” is set to true, or linear if set to false.

Cubic interpolation tends to follow the curves better, but linear is faster (and often, precise enough).

- Vector2 interpolatef(float fofs) const

Returns the position at the vertex “fofs”. It calls interpolate using the integer part of fofs as “idx”, and its fractional part as “t”.

- void remove_point(int idx)

Deletes the point “idx” from the curve. Sends an error to the console if “idx” is out of bounds.

- void set_bake_interval(float distance)
Sets the distance in pixels between two adjacent cached points. Changing it forces the cache to be recomputed the next time a xxx_baked_xxx function is called. The less distance, the more points the cache will have, and the more memory it will consume, so use with care.

- **void set_point_in ( int idx, Vector2 pos )**
  
  Sets the position of the control point leading to the vertex “idx”. If the index is out of bounds, the function sends an error to the console.

- **void set_point_out ( int idx, Vector2 pos )**
  
  Sets the position of the control point leading out of the vertex “idx”. If the index is out of bounds, the function sends an error to the console.

- **void set_point_pos ( int idx, Vector2 pos )**
  
  Sets the position for the vertex “idx”. If the index is out of bounds, the function sends an error to the console.

- **Vector2Array tesselate ( int max_stages=5, float tolerance_degrees=4 ) const**
  
  Returns a list of points along the curve, with a curvature controlled point density. That is, the curvier parts will have more points than the straighter parts.

  This approximation makes straight segments between each point, then subdivides those segments until the resulting shape is similar enough.

  “max_stages” controls how many subdivisions a curve segment may face before it is considered approximate enough. Each subdivision splits the segment in half, so the default 5 stages may mean up to 32 subdivisions per curve segment. Increase with care!

  “tolerance_degrees” controls how many degrees the midpoint of a segment may deviate from the real curve, before the segment has to be subdivided.

### 12.74 Curve3D

**Inherits:** *Resource < Reference < Object*

**Category:** Core

#### 12.74.1 Brief Description

Describes a Bezier curve in 3D space.
12.74.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>add_point(Vector3 pos, Vector3 in=Vector3(0, 0, 0), Vector3 out=Vector3(0, 0, 0), int atpos=-1)</td>
<td>Adds a point to a curve, at position “pos”, with control points “in” and “out”. If “atpos” is given, the point is inserted before the point number “atpos”, moving that point (and every point after) after the inserted point. If “atpos” is not given, or is an illegal value (atpos &lt;0 or atpos &gt;= get_point_count), the point will be appended at the end of the point list.</td>
</tr>
<tr>
<td>void</td>
<td>clear_points()</td>
<td></td>
</tr>
<tr>
<td>float</td>
<td>get_bake_interval() const</td>
<td>Returns the distance between two adjacent cached points.</td>
</tr>
<tr>
<td>float</td>
<td>get_baked_length() const</td>
<td>Returns the total length of the curve, based on the cached points. Given enough density (see set_bake_interval), it should be approximate enough.</td>
</tr>
<tr>
<td>Vector3Array</td>
<td>get_baked_points() const</td>
<td>Returns the cache of points as a Vector3Array.</td>
</tr>
<tr>
<td>RealArray</td>
<td>get_baked_tilts() const</td>
<td></td>
</tr>
<tr>
<td>Vector3</td>
<td>get_point_in(int idx) const</td>
<td></td>
</tr>
<tr>
<td>Vector3</td>
<td>get_point_out(int idx) const</td>
<td></td>
</tr>
<tr>
<td>Vector3</td>
<td>get_point_pos(int idx) const</td>
<td></td>
</tr>
<tr>
<td>float</td>
<td>get_point_tilt(int idx) const</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>remove_point(int idx)</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>set_bake_interval(float distance)</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>set_point_in(int idx, Vector3 pos)</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>set_point_out(int idx, Vector3 pos)</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>set_point_pos(int idx, Vector3 pos)</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>set_point_tilt(int idx, float tilt)</td>
<td></td>
</tr>
<tr>
<td>Vector3Array</td>
<td>tesselate(int max_stages=5, float tolerance_degrees=4) const</td>
<td></td>
</tr>
</tbody>
</table>

12.74.3 Description

This class describes a Bezier curve in 3D space. It is mainly used to give a shape to a Path, but can be manually sampled for other purposes.

It keeps a cache of precalculated points along the curve, to speed further calculations up.

12.74.4 Member Function Description

- void add_point(Vector3 pos, Vector3 in=Vector3(0, 0, 0), Vector3 out=Vector3(0, 0, 0), int atpos=-1)

Adds a point to a curve, at position “pos”, with control points “in” and “out”.

If “atpos” is given, the point is inserted before the point number “atpos”, moving that point (and every point after) after the inserted point. If “atpos” is not given, or is an illegal value (atpos <0 or atpos >= get_point_count), the point will be appended at the end of the point list.

- void clear_points()

Returns the distance between two adjacent cached points.

- float get_bake_interval() const

Returns the total length of the curve, based on the cached points. Given enough density (see set_bake_interval), it should be approximate enough.

- Vector3Array get_baked_points() const

Returns the cache of points as a Vector3Array.

- RealArray get_baked_tilts() const
Returns the cache of tilts as a `RealArray`.

- `int get_point_count()` const

Returns the number of points describing the curve.

- `Vector3 get_point_in(int idx)` const

Returns the position of the control point leading to the vertex “idx”. If the index is out of bounds, the function sends an error to the console, and returns (0, 0, 0).

- `Vector3 get_point_out(int idx)` const

Returns the position of the control point leading out of the vertex “idx”. If the index is out of bounds, the function sends an error to the console, and returns (0, 0, 0).

- `Vector3 get_point_pos(int idx)` const

Returns the position of the vertex “idx”. If the index is out of bounds, the function sends an error to the console, and returns (0, 0, 0).

- `float get_point_tilt(int idx)` const

Returns the tilt angle in radians for the point “idx”. If the index is out of bounds, the function sends an error to the console, and returns 0.

- `Vector3 interpolate(int idx, float t)` const

Returns the position between the vertex “idx” and the vertex “idx”+1, where “t” controls if the point is the first vertex (t = 0.0), the last vertex (t = 1.0), or in between. Values of “t” outside the range (0.0 >= t <= 1) give strange, but predictable results.

If “idx” is out of bounds it is truncated to the first or last vertex, and “t” is ignored. If the curve has no points, the function sends an error to the console, and returns (0, 0, 0).

- `Vector3 interpolate_baked(float offset, bool cubic=false)` const

Returns a point within the curve at position “offset”, where “offset” is measured as a distance in 3D units along the curve.

To do that, it finds the two cached points where the “offset” lies between, then interpolates the values. This interpolation is cubic if “cubic” is set to true, or linear if set to false.

Cubic interpolation tends to follow the curves better, but linear is faster (and often, precise enough).

- `Vector3 interpolatef(float fofs)` const

Returns the position at the vertex “fofs”. It calls `interpolate` using the integer part of fofs as “idx”, and its fractional part as “t”.

- `void remove_point(int idx)`

Deletes the point “idx” from the curve. Sends an error to the console if “idx” is out of bounds.

- `void set_bake_interval(float distance)`

Sets the distance in 3D units between two adjacent cached points. Changing it forces the cache to be recomputed the next time a xxx_baked_xxx function is called. The less distance, the more points the cache will have, and the more memory it will consume, so use with care.

- `void set_point_in(int idx, Vector3 pos)`

Sets the position of the control point leading to the vertex “idx”. If the index is out of bounds, the function sends an error to the console.

- `void set_point_out(int idx, Vector3 pos)`
Sets the position of the control point leading out of the vertex “idx”. If the index is out of bounds, the function sends an error to the console.

- `void set_point_pos ( int idx, Vector3 pos )`

Sets the position for the vertex “idx”. If the index is out of bounds, the function sends an error to the console.

- `void set_point_tilt ( int idx, float tilt )`

Sets the tilt angle in radians for the point “idx”. If the index is out of bounds, the function sends an error to the console.

The tilt controls the rotation along the look-at axis an object traveling the path would have. In the case of a curve controlling a `PathFollow`, this tilt is an offset over the natural tilt the `PathFollow` calculates.

- `Vector3Array tesselate ( int max_stages=5, float tolerance_degrees=4 ) const`

Returns a list of points along the curve, with a curvature controlled point density. That is, the curvier parts will have more points than the straighter parts.

This approximation makes straight segments between each point, then subdivides those segments until the resulting shape is similar enough.

“max_stages” controls how many subdivisions a curve segment may face before it is considered approximate enough. Each subdivision splits the segment in half, so the default 5 stages may mean up to 32 subdivisions per curve segment. Increase with care!

“tolerance_degrees” controls how many degrees the midpoint of a segment may deviate from the real curve, before the segment has to be subdivided.

### 12.75 DampedSpringJoint2D

**Inherits:** Joint2D < Node2D < CanvasItem < Node < Object

**Category:** Core

#### 12.75.1 Brief Description

Damped spring constraint for 2D physics.

#### 12.75.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>float</td>
<td>get_damping ( ) const</td>
</tr>
<tr>
<td>float</td>
<td>get_length ( ) const</td>
</tr>
<tr>
<td>float</td>
<td>get_rest_length ( ) const</td>
</tr>
<tr>
<td>float</td>
<td>get_stiffness ( ) const</td>
</tr>
<tr>
<td>void</td>
<td>set_damping ( float damping )</td>
</tr>
<tr>
<td>void</td>
<td>set_length ( float length )</td>
</tr>
<tr>
<td>void</td>
<td>set_rest_length ( float rest_length )</td>
</tr>
<tr>
<td>void</td>
<td>set_stiffness ( float stiffness )</td>
</tr>
</tbody>
</table>

#### 12.75.3 Description

Damped spring constraint for 2D physics. This resembles a spring joint that always wants to go back to a given length.
12.75.4 Member Function Description

- `float get_damping() const`
  Return the damping ratio of the spring joint. A value of 0 indicates an undamped spring, while 1 causes the system to reach equilibrium as fast as possible (critical damping).

- `float get_length() const`
  Return the maximum length of the spring joint.

- `float get_rest_length() const`
  Return the resting length of the spring joint. The joint will always try to go back to this length when pulled apart.

- `float get_stiffness() const`
  Return the stiffness of the spring joint. The joint applies a force equal to the stiffness times the distance from its resting length.

- `void set_damping(float damping)`
  Set the damping ratio of the spring joint. A value of 0 indicates an undamped spring, while 1 causes the system to reach equilibrium as fast as possible (critical damping).

- `void set_length(float length)`
  Set the maximum length of the spring joint.

- `void set_rest_length(float rest_length)`
  Set the resting length of the spring joint. The joint will always try to go back to this length when pulled apart.

- `void set_stiffness(float stiffness)`
  Set the stiffness of the spring joint. The joint applies a force equal to the stiffness times the distance from its resting length.

12.76 Dictionary

Category: Built-In Types

12.76.1 Brief Description

Dictionary type.
12.76.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>clear()</td>
</tr>
<tr>
<td>bool</td>
<td>empty()</td>
</tr>
<tr>
<td>void</td>
<td>erase(var key)</td>
</tr>
<tr>
<td>bool</td>
<td>has(var key)</td>
</tr>
<tr>
<td>bool</td>
<td>has_all(Array keys)</td>
</tr>
<tr>
<td>int</td>
<td>hash()</td>
</tr>
<tr>
<td>Array</td>
<td>keys()</td>
</tr>
<tr>
<td>int</td>
<td>parse_json(String json)</td>
</tr>
<tr>
<td>int</td>
<td>size()</td>
</tr>
<tr>
<td>String</td>
<td>to_json()</td>
</tr>
<tr>
<td>Array</td>
<td>values()</td>
</tr>
</tbody>
</table>

12.76.3 Description

Dictionary type. Associative container which contains values referenced by unique keys. Dictionaries are always passed by reference.

12.76.4 Member Function Description

- void clear()
  Clear the dictionary, removing all key/value pairs.

- bool empty()
  Return true if the dictionary is empty.

- void erase(var key)
  Erase a dictionary key/value pair by key.

- bool has(var key)
  Return true if the dictionary has a given key.

- bool has_all(Array keys)
  Return true if the dictionary has all of the keys in the given array.

- int hash()
  Return a hashed integer value representing the dictionary contents.

- Array keys()
  Return the list of keys in the Dictionary.

- int parse_json(String json)
  Parse JSON text to the dictionary. Return OK when succeeded or the error code when failed.

Be aware that the JSON specification does not define integer or float types, but only a number type. Therefore, parsing a JSON text will convert every numerical values to float types.

- int size()
  Return the size of the dictionary (in pairs).

- String to_json()
  Return the JSON representation of the dictionary.
Return the dictionary as json text.

- Array values ()

Return the list of values in the Dictionary.

12.77 DirectionalLight

Inherits: Light < VisualInstance < Spatial < Node < Object

Category: Core

12.77.1 Brief Description

Directional Light, such as the Sun or the Moon.

12.77.2 Member Functions

```
<table>
<thead>
<tr>
<th>Function</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>int get_shadow_mode() const</td>
<td></td>
</tr>
<tr>
<td>float get_shadow_param(int param) const</td>
<td></td>
</tr>
<tr>
<td>void set_shadow_mode(int mode)</td>
<td></td>
</tr>
<tr>
<td>void set_shadow_param(int param, float value)</td>
<td></td>
</tr>
</tbody>
</table>
```

12.77.3 Numeric Constants

- SHADOW_ORTHOGONAL = 0
- SHADOW_PERSPECTIVE = 1
- SHADOW_PARALLEL_2_SPLITS = 2
- SHADOW_PARALLEL_4_SPLITS = 3
- SHADOW_PARAM_MAX_DISTANCE = 0
- SHADOW_PARAM_PSSM_SPLIT_WEIGHT = 1
- SHADOW_PARAM_PSSM_ZOFFSET_SCALE = 2

12.77.4 Description

A DirectionalLight is a type of Light node that emits light constantly in one direction (the negative z axis of the node). It is used lights with strong intensity that are located far away from the scene to model sunlight or moonlight. The worldspace location of the DirectionalLight transform (origin) is ignored, only the basis is used do determine light direction.

12.77.5 Member Function Description

- int get_shadow_mode() const
- float get_shadow_param(int param) const
- void set_shadow_mode(int mode)
• void set_shadow_param ( int param, float value )

12.78 Directory

Inherits: Reference < Object

Category: Core

12.78.1 Brief Description

Type used to handle the filesystem.

12.78.2 Member Functions

<table>
<thead>
<tr>
<th>Error</th>
<th>change_dir ( String todir )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error</td>
<td>copy ( String from, String to )</td>
</tr>
<tr>
<td>bool</td>
<td>current_is_dir ( ) const</td>
</tr>
<tr>
<td>bool</td>
<td>dir_exists ( String path )</td>
</tr>
<tr>
<td>bool</td>
<td>file_exists ( String path )</td>
</tr>
<tr>
<td>String</td>
<td>get_current_dir ( )</td>
</tr>
<tr>
<td>String</td>
<td>get_drive ( int idx )</td>
</tr>
<tr>
<td>int</td>
<td>get_drive_count ( )</td>
</tr>
<tr>
<td>String</td>
<td>get_next ( )</td>
</tr>
<tr>
<td>int</td>
<td>get_space_left ( )</td>
</tr>
<tr>
<td>bool</td>
<td>list_dir_begin ( )</td>
</tr>
<tr>
<td>void</td>
<td>list_dir_end ( )</td>
</tr>
<tr>
<td>Error</td>
<td>make_dir ( String path )</td>
</tr>
<tr>
<td>Error</td>
<td>make_dir_recursive ( String path )</td>
</tr>
<tr>
<td>Error</td>
<td>open ( String path )</td>
</tr>
<tr>
<td>Error</td>
<td>remove ( String path )</td>
</tr>
<tr>
<td>Error</td>
<td>rename ( String from, String to )</td>
</tr>
</tbody>
</table>

12.78.3 Description

Directory type. It is used to manage directories and their content (not restricted to the project folder).

Here is an example on how to iterate through the files of a directory:

```python
func dir_contents(path):
    var dir = Directory.new()
    if dir.open(path) == OK:
        dir.list_dir_begin()
        var file_name = dir.get_next()
        while (file_name != ""):
            if dir.current_is_dir():
                print("Found directory: " + file_name)
            else:
                print("Found file: " + file_name)
                file_name = dir.get_next()
        else:
            print("An error occurred when trying to access the path.")
    else:
        print("An error occurred when trying to access the path.")
```
### 12.78.4 Member Function Description

- **Error** `change_dir (String todir)`

  Change the currently opened directory to the one passed as an argument. The argument can be relative to the current directory (e.g. `newdir` or `../newdir`), or an absolute path (e.g. `/tmp/newdir` or `res://somedir/newdir`). The method returns one of the error code constants defined in `@Global Scope` (OK or ERR_*)..

- **Error** `copy (String from, String to)`

  Copy the `from` file to the `to` destination. Both arguments should be paths to files, either relative or absolute. If the destination file exists and is not access-protected, it will be overwritten.

  Returns one of the error code constants defined in `@Global Scope` (OK, FAILED or ERR_*).

- **bool** `current_is_dir ()` const

  Return whether the current item processed with the last `get_next` call is a directory (.. and .. are considered directories).

- **bool** `dir_exists (String path)`

  Return whether the target directory exists. The argument can be relative to the current directory, or an absolute path.

- **bool** `file_exists (String path)`

  Return whether the target file exists. The argument can be relative to the current directory, or an absolute path.

- **String** `get_current_dir ()`

  Return the absolute path to the currently opened directory (e.g. `res://folder` or `C:\tmp\folder`).

- **String** `get_drive (int idx)`

  On Windows, return the name of the drive (partition) passed as an argument (e.g. `C:`). On other platforms, or if the requested drive does not existed, the method returns an empty String.

- **int** `get_drive_count ()`

  On Windows, return the number of drives (partitions) mounted on the current filesystem. On other platforms, the method returns 0.

- **String** `get_next ()`

  Return the next element (file or directory) in the current directory (including .. and ..). The name of the file or directory is returned (and not its full path). Once the stream has been fully processed, the method returns an empty String and closes the stream automatically (i.e. `list_dir_end` would not be mandatory in such a case).

- **int** `get_space_left ()`

  On Unix desktop systems, return the available space on the current directory's disk. On other platforms, this information is not available and the method returns 0 or -1.

- **bool** `list_dir_begin ()`

  Initialise the stream used to list all files and directories using the `get_next` function, closing the current opened stream if needed. Once the stream has been processed, it should typically be closed with `list_dir_end`.

  Return true if the stream could not be initialised.

- **void** `list_dir_end ()`

  Close the current stream opened with `list_dir_begin` (whether it has been fully processed with `get_next` or not does not matter).

- **Error** `make_dir (String path)`
Create a directory. The argument can be relative to the current directory, or an absolute path. The target directory should be placed in an already existing directory (to create the full path recursively, see `make_dir_recursive`). The method returns one of the error code constants defined in `@Global Scope` (OK, FAILED or ERR_*).

- Error `make_dir_recursive (String path)`
  
  Create a target directory and all necessary intermediate directories in its path, by calling `make_dir` recursively. The argument can be relative to the current directory, or an absolute path.

  Return one of the error code constants defined in `@Global Scope` (OK, FAILED or ERR_*).

- Error `open (String path)`
  
  Open an existing directory of the filesystem. The `path` argument can be within the project tree (`res://folder`), the user directory (`user://folder`) or an absolute path of the user filesystem (e.g. `/tmp/folder` or `C:\tmp\folder`).

  The method returns one of the error code constants defined in `@Global Scope` (OK or ERR_*).

- Error `remove (String path)`
  
  Delete the target file or an empty directory. The argument can be relative to the current directory, or an absolute path. If the target directory is not empty, the operation will fail.

  Return one of the error code constants defined in `@Global Scope` (OK or FAILED).

- Error `rename (String from, String to)`
  
  Rename (move) the `from` file to the `to` destination. Both arguments should be paths to files, either relative or absolute. If the destination file exists and is not access-protected, it will be overwritten.

  Return one of the error code constants defined in `@Global Scope` (OK or FAILED).

### 12.79 DynamicFont

Inherits: `Font < Resource < Reference < Object`

Category: Core
12.79.1 Brief Description

12.79.2 Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>add_fallback</td>
<td>void add_fallback ( DynamicFontData data )</td>
</tr>
<tr>
<td>get_fallback</td>
<td>DynamicFontData get_fallback ( int idx ) const</td>
</tr>
<tr>
<td>get_fallback_count</td>
<td>int get_fallback_count ( ) const</td>
</tr>
<tr>
<td>get_font_data</td>
<td>DynamicFontData get_font_data ( ) const</td>
</tr>
<tr>
<td>get_size</td>
<td>int get_size ( ) const</td>
</tr>
<tr>
<td>get_spacing</td>
<td>int get_spacing ( int type ) const</td>
</tr>
<tr>
<td>get_use_filter</td>
<td>bool get_use_filter ( ) const</td>
</tr>
<tr>
<td>get_use_mipmaps</td>
<td>bool get_use_mipmaps ( ) const</td>
</tr>
<tr>
<td>remove_fallback</td>
<td>void remove_fallback ( int idx )</td>
</tr>
<tr>
<td>set_fallback</td>
<td>void set_fallback ( int idx, DynamicFontData data )</td>
</tr>
<tr>
<td>set_font_data</td>
<td>void set_font_data ( DynamicFontData data )</td>
</tr>
<tr>
<td>set_size</td>
<td>void set_size ( int data )</td>
</tr>
<tr>
<td>set_spacing</td>
<td>void set_spacing ( int type, int value )</td>
</tr>
<tr>
<td>set_use_filter</td>
<td>void set_use_filter ( bool enable )</td>
</tr>
<tr>
<td>set_use_mipmaps</td>
<td>void set_use_mipmaps ( bool enable )</td>
</tr>
</tbody>
</table>

12.79.3 Numeric Constants

- SPACING_TOP = 0
- SPACING_BOTTOM = 1
- SPACING_CHAR = 2
- SPACING_SPACE = 3

12.79.4 Member Function Description

- void add_fallback ( DynamicFontData data )
- DynamicFontData get_fallback ( int idx ) const
- int get_fallback_count ( ) const
- DynamicFontData get_font_data ( ) const
- int get_size ( ) const
- int get_spacing ( int type ) const
- bool get_use_filter ( ) const
- bool get_use_mipmaps ( ) const
- void remove_fallback ( int idx )
- void set_fallback ( int idx, DynamicFontData data )
- void set_font_data ( DynamicFontData data )
- void set_size ( int data )
- void set_spacing ( int type, int value )
- void set_use_filter ( bool enable )
12.80 DynamicFontData

Inherits: Resource < Reference < Object
Category: Core

12.80.1 Brief Description

12.80.2 Member Functions

<table>
<thead>
<tr>
<th>String</th>
<th>get_font_path ( ) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>set_font_path ( String path )</td>
</tr>
</tbody>
</table>

12.80.3 Member Function Description

- String get_font_path ( ) const
- void set_font_path ( String path )

12.81 EditorExportPlugin

Inherits: Reference < Object
Category: Core

12.81.1 Brief Description

Editor plugin to control the export process.

12.81.2 Member Functions

| Variant | custom_export ( String name, EditorExportPlatform platform ) virtual |

12.81.3 Description

This plugin is added into EditorImportExport and allows to modify the behavior of the export process for individual files.

12.81.4 Member Function Description

- Variant custom_export ( String name, EditorExportPlatform platform ) virtual

This function is called for each file exported and depending from the return value one of many things might happen.

1. If returned value is null, the file is exported as is.

12.80. DynamicFontData
2. If the returned value is a RawArray (array of bytes), the content of that array becomes the new file being exported.

3. If the file must also change its name when exported, then a Dictionary must be returned with two fields: ‘name’ with the new filename and ‘data’ with a RawArray containing the raw contents of the file. Even if the name is changed, the run-time will redirect the old file to the new file automatically when accessed.

12.82 EditorFileDialog

Inherits: ConfirmationDialog << AcceptDialog << WindowDialog << Popup << Control << CanvasItem << Node << Object
Category: Core

12.82.1 Brief Description

12.82.2 Member Functions

<table>
<thead>
<tr>
<th>void</th>
<th>add_filter ( String filter )</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>clear_filters ( )</td>
</tr>
<tr>
<td>int</td>
<td>get_access ( ) const</td>
</tr>
<tr>
<td>String</td>
<td>get_current_dir ( ) const</td>
</tr>
<tr>
<td>String</td>
<td>get_current_file ( ) const</td>
</tr>
<tr>
<td>String</td>
<td>get_current_path ( ) const</td>
</tr>
<tr>
<td>int</td>
<td>get_display_mode ( ) const</td>
</tr>
<tr>
<td>int</td>
<td>get_mode ( ) const</td>
</tr>
<tr>
<td>VBoxContainer</td>
<td>get_vbox ( )</td>
</tr>
<tr>
<td>void</td>
<td>invalidate ( )</td>
</tr>
<tr>
<td>bool</td>
<td>is_overwrite_warning_disabled ( ) const</td>
</tr>
<tr>
<td>bool</td>
<td>is_showing_hidden_files ( ) const</td>
</tr>
<tr>
<td>void</td>
<td>set_access ( int access )</td>
</tr>
<tr>
<td>void</td>
<td>set_current_dir ( String dir )</td>
</tr>
<tr>
<td>void</td>
<td>set_current_file ( String file )</td>
</tr>
<tr>
<td>void</td>
<td>set_current_path ( String path )</td>
</tr>
<tr>
<td>void</td>
<td>set_disable_overwrite_warning ( bool disable )</td>
</tr>
<tr>
<td>void</td>
<td>set_display_mode ( int mode )</td>
</tr>
<tr>
<td>void</td>
<td>set_mode ( int mode )</td>
</tr>
<tr>
<td>void</td>
<td>set_show_hidden_files ( bool show )</td>
</tr>
</tbody>
</table>

12.82.3 Signals

- dir_selected ( String dir )
- file_selected ( String path )
- files_selected ( StringArray paths )

12.82.4 Numeric Constants

- MODE_OPEN_FILE = 0
- MODE_OPEN_FILES = 1
- MODE_OPEN_DIR = 2
• MODE_OPEN_ANY = 3
• MODE_SAVE_FILE = 4
• ACCESS_RESOURCES = 0
• ACCESS_USERDATA = 1
• ACCESS_FILESYSTEM = 2

12.82.5 Member Function Description

• void add_filter (String filter)
• void clear_filters()
• int get_access() const
• String get_current_dir() const
• String get_current_file() const
• String get_current_path() const
• int get_display_mode() const
• int get_mode() const
• VBoxContainer get_vbox()
• void invalidate()
• bool is_overwrite_warning_disabled() const
• bool is_showing_hidden_files() const
• void set_access(int access)
• void set_current_dir(String dir)
• void set_current_file(String file)
• void set_current_path(String path)
• void set_disable_overwrite_warning(bool disable)
• void set_display_mode(int mode)
• void set_mode(int mode)
• void set_show_hidden_files(bool show)

12.83 EditorImportPlugin

Inherits: Reference < Object
Category: Core

12.83.1 Brief Description

Import plugin for editor
12.83.2 Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>bool can_reimport_multiple_files()</code> virtual</td>
<td>Determines if multiple files can be re-imported.</td>
</tr>
<tr>
<td><code>RawArray custom_export(String path, EditorExportPlatform platform)</code> virtual</td>
<td>Customizes the export process.</td>
</tr>
<tr>
<td><code>String expand_source_path(String path)</code></td>
<td>Expands the source path.</td>
</tr>
<tr>
<td><code>String get_name()</code> virtual</td>
<td>Returns the name of the import plugin.</td>
</tr>
<tr>
<td><code>String get_visible_name()</code> virtual</td>
<td>Returns the visible name for this plugin.</td>
</tr>
<tr>
<td><code>int import(String path, ResourceImportMetadata from)</code> virtual</td>
<td>Performs an import of an external resource into the project.</td>
</tr>
<tr>
<td><code>void import_dialog(String from)</code> virtual</td>
<td>Imports content from a dialog box.</td>
</tr>
<tr>
<td><code>void import_from_drop(StringArray files, String dest_path)</code> virtual</td>
<td>Imports content from a drop event.</td>
</tr>
<tr>
<td><code>void reimport_multiple_files(StringArray files)</code> virtual</td>
<td>Re-imports multiple files.</td>
</tr>
<tr>
<td><code>String validate_source_path(String path)</code></td>
<td>Validates the source path.</td>
</tr>
</tbody>
</table>

12.83.3 Description

Import plugins make it easy to handle importing of external assets into a project. The way they work is not that obvious though, so please make sure to read the documentation, tutorials and examples.

12.83.4 Member Function Description

- `bool can_reimport_multiple_files()` virtual
- `RawArray custom_export(String path, EditorExportPlatform platform)` virtual

Generally, files that are imported stay the same when exported. The only exception is in some cases when the file must be re-imported for different platforms (i.e. texture compression).

If you want to customize the export process, it’s recommended to use `EditorExportPlugin.custom_export` instead.

- `String expand_source_path(String path)`
- `String get_name()` virtual

Get the name of the import plugin, which will be used to identify content imported by this plugin. Try to use lowercase and underscores if possible.

- `String get_visible_name()` virtual

Visible name for this plugin, which will be shown on the import menu.

- `int import(String path, ResourceImportMetadata from)` virtual

Perform an import of an external resource into the project. This function is both called on import (from the dialog) or re-import (manual or automatic when external source files changed).

An import process generally works like this:

1. Check the metadata for source files and options. Metadata is either generated in the import dialog or taken from an existing resource upon reimport.
2. Perform the import process into a new resource. Some times the resource being re-imported may be already loaded and in use, so checking for this by using `ResourceLoader.has` is recommended. Otherwise create a new resource.
3. Set the metadata from the argument into the existing or new resource being created using `Resource.set_import_metadata`.
4. Save the resource into ‘path’ (function argument)

- `void import_dialog(String from)` virtual
This function is called when either the user chooses to import a resource of this type (Import menu), or when the user chooses to re-import the resource (from filesystem). In the later case, the path for the existing file is supplied in the argument.

If the path is supplied, it is recommended to read the import metadata with ResourceLoader.load_import_metadata and fill in the fields with the values contained there.

The dialog can be shown in any way (just use a ConfirmationDialog and pop it up). Upon confirmation, fill up a ResourceImportMetadata and call the EditorImportPlugin.import function with this information.

- void import_from_drop (StringArray files, String dest_path) virtual
- void reimport_multiple_files (StringArray files) virtual
- String validate_source_path (String path)

## 12.84 EditorPlugin

**Inherits:** Node < Object  
**Category:** Core

### 12.84.1 Brief Description

Used by the editor to extend its functionality.

### 12.84.2 Member Functions

<table>
<thead>
<tr>
<th>Member Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>add_control_to_bottom_panel</td>
<td>Control control, String title</td>
</tr>
<tr>
<td>add_control_to_container</td>
<td>int container, Control control</td>
</tr>
<tr>
<td>add_control_to_dock</td>
<td>int slot, Control control</td>
</tr>
<tr>
<td>add_custom_type</td>
<td>String type, String base, Script script, Texture icon</td>
</tr>
<tr>
<td>add_export_plugin</td>
<td>EditorExportPlugin plugin</td>
</tr>
<tr>
<td>add_import_plugin</td>
<td>EditorImportPlugin plugin</td>
</tr>
<tr>
<td>apply_changes</td>
<td>virtual</td>
</tr>
<tr>
<td>clear</td>
<td>virtual</td>
</tr>
<tr>
<td>create_spatial_gizmo</td>
<td>Spatial for_spatial</td>
</tr>
<tr>
<td>edit</td>
<td>Object object</td>
</tr>
<tr>
<td>edit_resource</td>
<td>Object arg0</td>
</tr>
<tr>
<td>forward_input_event</td>
<td>InputEvent event</td>
</tr>
<tr>
<td>forward.spatial_input_event</td>
<td>Camera camera, InputEvent event</td>
</tr>
<tr>
<td>get_base_control</td>
<td>()</td>
</tr>
<tr>
<td>get_breakpoints</td>
<td>() virtual</td>
</tr>
<tr>
<td>get_editor_settings</td>
<td>()</td>
</tr>
<tr>
<td>get_editor_viewport</td>
<td>()</td>
</tr>
<tr>
<td>get_name</td>
<td>() virtual</td>
</tr>
<tr>
<td>get_selection</td>
<td>()</td>
</tr>
<tr>
<td>get_state</td>
<td>() virtual</td>
</tr>
<tr>
<td>get_undo_redo</td>
<td>()</td>
</tr>
<tr>
<td>get_window_layout</td>
<td>ConfigFile layout</td>
</tr>
<tr>
<td>handles</td>
<td>Object object</td>
</tr>
</tbody>
</table>

Continued on next page
### Table 12.13 – continued from previous page

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool  <code>has_main_screen()</code> virtual</td>
<td></td>
</tr>
<tr>
<td>void  <code>make_visible(bool visible)</code> virtual</td>
<td></td>
</tr>
<tr>
<td>void  <code>queue_save_layout()</code> const</td>
<td></td>
</tr>
<tr>
<td>void  <code>remove_control_from_bottom_panel(Control control)</code></td>
<td></td>
</tr>
<tr>
<td>void  <code>remove_control_from_docks(Control control)</code></td>
<td></td>
</tr>
<tr>
<td>void  <code>remove_custom_type(String type)</code></td>
<td></td>
</tr>
<tr>
<td>void  <code>remove_export_plugin(EditorExportPlugin plugin)</code></td>
<td></td>
</tr>
<tr>
<td>void  <code>remove_import_plugin(EditorImportPlugin plugin)</code></td>
<td></td>
</tr>
<tr>
<td>void  <code>save_external_data()</code> virtual</td>
<td></td>
</tr>
<tr>
<td>void  <code>set_state(Dictionary state)</code> virtual</td>
<td></td>
</tr>
<tr>
<td>void  <code>set_window_layout(ConfigFile layout)</code> virtual</td>
<td></td>
</tr>
</tbody>
</table>

### 12.84.3 Numeric Constants

- CONTAINER_TOOLBAR = 0
- CONTAINER_SPATIAL_EDITOR_MENU = 1
- CONTAINER_SPATIAL_EDITOR_SIDE = 2
- CONTAINER_SPATIAL_EDITOR_BOTTOM = 3
- CONTAINER_CANVAS_EDITOR_MENU = 4
- CONTAINER_CANVAS_EDITOR_SIDE = 5
- CONTAINER_PROPERTY_EDITOR_BOTTOM = 7
- DOCK_SLOT_LEFT_UL = 0
- DOCK_SLOT_LEFT_BL = 1
- DOCK_SLOT_LEFT_UR = 2
- DOCK_SLOT_LEFT_BR = 3
- DOCK_SLOT_RIGHT_UL = 4
- DOCK_SLOT_RIGHT_BL = 5
- DOCK_SLOT_RIGHT_UR = 6
- DOCK_SLOT_RIGHT_BR = 7
- DOCK_SLOT_MAX = 8

### 12.84.4 Description

Plugins are used by the editor to extend functionality. The most common types of plugins are those which edit a given node or resource type, import plugins and export plugins.

### 12.84.5 Member Function Description

- `ToolButton add_control_to_bottom_panel(Control control, String title)`

Add a control to the bottom panel (together with Output, Debug, Animation, etc). Returns a reference to the button added. It’s up to you to hide/show the button when needed. If your plugin is being removed, also make sure to remove your control by calling `remove_control_from_bottom_panel`. 
• void add_control_to_container (int container, Control control)

Add a custom control to a container (see CONTAINER_* enum). There are many locations where custom controls can be added in the editor UI.

Please remember that you have to manage the visibility of your custom controls yourself (and likely hide it after adding it).

If your plugin is being removed, also make sure to remove your custom controls too.

• void add_control_to_dock (int slot, Control control)

Add the control to a specific dock slot (see DOCK_* enum for options).

If the dock is repositioned and as long as the plugin is active, the editor will save the dock position on further sessions.

If your plugin is being removed, also make sure to remove your control by calling remove_control_from_docks.

• void add_custom_type (String type, String base, Script script, Texture icon)

Add a custom type, which will appear in the list of nodes or resources. An icon can be optionally passed.

When given node or resource is selected, the base type will be instanced (ie, “Spatial”, “Control”, “Resource”), then the script will be loaded and set to this object.

You can use the EditorPlugin.handles to check if your custom object is being edited by checking the script or using ‘extends’ keyword.

During run-time, this will be a simple object with a script so this function does not need to be called then.

• void add_export_plugin (EditorExportPlugin plugin)

Add an export plugin. Plugins of this kind can change files being exported. On exit don’t forget to call remove_export_plugin.

• void add_import_plugin (EditorImportPlugin plugin)

Add an import plugin. These plugins manage importing external content (from outside the project) into formats the engine can understand.

On exit, don’t forget to remove the plugin by calling remove_import_plugin.

• void apply_changes ( ) virtual

This method is called when the editor is about to save the project, switch to another tab, etc. It asks the plugin to apply any pending state changes to ensure consistency.

This is used, for example, in shader editors to let the plugin know that it must apply the shader code being written by the user to the object.

• void clear ( ) virtual

Clear all the state and reset the object being edited to zero. This ensures your plugin does not keep editing a currently existing node, or a node from the wrong scene.

• EditorSpatialGizmo create_spatial_gizmo (Spatial for_spatial) virtual

This is used for plugins that create gizmos used by the spatial editor. Just check that the node passed in the “for_spatial” argument matches your plugin.

• void edit (Object object) virtual

This function is used for plugins that edit specific object types (nodes or resources). It requests the editor to edit the given object.

• void edit_resource (Object arg0)
Tells the editor to handle the edit of the given resource. For example, if you pass a script as argument, the editor will open the script editor.

- **bool forward_input_event (InputEvent event) virtual**
  
  This is a low level function for plugins that edit a given object type derived from CanvasItem to capture the input in the 2D editor viewport. The function is only being called if your object is being edited.

  Return true if you want to capture the input, otherwise false.

- **bool forward_spatial_input_event (Camera camera, InputEvent event) virtual**
  
  This is a low level function for plugins that edit a given object type derived from Spatial to capture the input of the viewport. The function is only being called if your object is being edited.

  By using the *InputEvent* and the *Camera* arguments it’s pretty easy to do raycasts into space using Camera functions.

  Return true if you want to capture the input, otherwise false.

- **Control get_base_control ()**
  
  Get a base control where it’s safe to place dialogs. Many plugins open dialogs and they need a control as a base to make sure they use the editor icons and theme.

- **StringArray get_breakpoints () virtual**
  
  This is for editors that edit script based objects. You can return a list of breakpoints in the format (script:line), for example: res://path_to_script.gd:25

  - **EditorSettings get_editor_settings ()**
  
    Get the general settings for the editor (the same window that appears in the Settings menu).

  - **Control get_editor_viewport ()**
  
    Get the main editor control. Use this as a parent for main screens.

  - **String get_name () virtual**
  
    Get the name of the editor plugin. For main screen plugins this is what will appear in the selector (which by default is 2D, 3D, Script).

  - **EditorSelection get_selection ()**
  
    Get the object that handles the selection of nodes in the Scene Tree editor.

  - **Dictionary get_state () virtual**
  
    Get the state of your plugin editor. This is used when saving the scene (so state is kept when opening it again) and for switching tabs (so state can be restored when the tab returns).

  - **UndoRedo get_undo_redo ()**
  
    Get the undo/redo object. Most actions in the editor can be undoable, so use this object to make sure this happens when it’s worth it.

  - **void get_window_layout (ConfigFile layout) virtual**
  
    Get the GUI layout of the plugin. This is used to save the project’s editor layout when the *EditorPlugin.queue_save_layout* is called or the editor layout was changed(For example changing the position of a dock).

  - **bool handles (Object object) virtual**
  
    Implement this function if your plugin edits a specific type of object (Resource or Node). If you return true, then you will get the functions *EditorPlugin.edit* and *EditorPlugin.make_visible* called when the editor requests them.

  - **bool has_main_screen () virtual**
Return true if this is a main screen editor plugin (it goes in the main screen selector together with 2D, 3D, Script).

- void make_visible ( bool visible ) virtual

This function will be called when the editor is requested to become visible. It is used for plugins that edit a specific object type.

Remember that you have to manage the visibility of all your editor controls manually.

- void queue_save_layout ( ) const

Queue save the project’s editor layout.

- void remove_control_from_bottom_panel ( Control control )

Remove the control from the bottom panel. Don’t forget to call this if you added one, so the editor can remove it cleanly.

- void remove_control_from_docks ( Control control )

Remove the control from the dock. Don’t forget to call this if you added one, so the editor can save the layout and remove it cleanly.

- void remove_custom_type ( String type )

Remove a custom type added by EditorPlugin.add_custom_type

- void remove_export_plugin ( EditorExportPlugin plugin )

Remove the export plugin, don’t forget to call this on exit.

- void remove_import_plugin ( EditorImportPlugin plugin )

Remove the import plugin, don’t forget to call this on exit.

- void save_external_data ( ) virtual

This method is called after the editor saves the project or when it’s closed. It asks the plugin to save edited external scenes/resources.

- void set_state ( Dictionary state ) virtual

Restore the state saved by EditorPlugin.get_state.

- void set_window_layout ( ConfigFile layout ) virtual

Restore the plugin GUI layout saved by EditorPlugin.get_window_layout.

### 12.85 EditorScenePostImport

**Inherits:** Reference < Object  

**Category:** Core

#### 12.85.1 Brief Description

Base script for post-processing scenes being imported.

#### 12.85.2 Member Functions

| void | post_import ( Object scene ) virtual |
12.85.3 Description

These scripts can modify scenes after being imported by the 3D Scene import option of the Import menu.

12.85.4 Member Function Description

- void **post_import** (Object scene ) virtual
  This function is called upon import with the imported scene.
  Just do any changes desired to the scene and return it. If null is returned, import will fail and throw an error to the user.

12.86 EditorScript

**Inherits:** Reference < Object  
**Category:** Core

12.86.1 Brief Description

Simple script to perform changes in the currently edited scene.

12.86.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>get_scene ( )</td>
</tr>
<tr>
<td>void</td>
<td>add_root_node ( Object node )</td>
</tr>
<tr>
<td>void</td>
<td>_run ( ) virtual</td>
</tr>
</tbody>
</table>

12.86.3 Description

This script can be run from the Scene -> Run Script menu option.

12.86.4 Member Function Description

- void **_run** ( ) virtual
- void **add_root_node** ( Object node )
- **Object get_scene** ( )

12.87 EditorSelection

**Inherits:** Object  
**Category:** Core
12.87.1 Brief Description

Manages the SceneTree selection in the editor.

12.87.2 Member Functions

| void   | add_node ( Node node )     |
| void   | clear ( )                  |
| Array  | get_selected_nodes ( )     |
| void   | remove_node ( Node node )  |

12.87.3 Signals

- selection_changed ( )

Emitted when the selection changes.

12.87.4 Description

This object manages the SceneTree selection in the editor.

12.87.5 Member Function Description

- void add_node ( Node node )
  Add a node to the selection.
  - void clear ( )
  Clear the selection.
  - Array get_selected_nodes ( )
  Get the list of selected nodes.
  - void remove_node ( Node node )
  Remove a node from the selection.

12.88 EditorSettings

Inherits: Resource < Reference < Object

Category: Core

12.88.1 Brief Description

Object that holds the project-independent editor settings.
12.88.2 Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void add_property_info(Dictionary info)</td>
<td>Add a custom property info to a property. The dictionary must contain: name:String, and optionally hint:int), hint_string:String.</td>
</tr>
<tr>
<td>void erase(String property)</td>
<td>Erase a given setting (pass full property path).</td>
</tr>
<tr>
<td>StringArray get_favorite_dirs() const</td>
<td>Get the list of favorite directories for this project.</td>
</tr>
<tr>
<td>String get_project_settings_path() const</td>
<td></td>
</tr>
<tr>
<td>StringArray get_recent_dirs() const</td>
<td></td>
</tr>
<tr>
<td>String get_settings_path() const</td>
<td></td>
</tr>
<tr>
<td>void set_favorite_dirs(StringArray dirs)</td>
<td></td>
</tr>
<tr>
<td>void set_recent_dirs(StringArray dirs)</td>
<td></td>
</tr>
</tbody>
</table>

12.88.3 Signals

- settings_changed()

12.88.4 Description

Object that holds the project-independent editor settings. These settings are generally visible in the Editor Settings menu.

Accessing the settings is done by using the regular Object API, such as:

```java
settings.set(prop, value)
settings.get(prop)
list_of_settings = settings.get_property_list()
```

12.88.5 Member Function Description

- void add_property_info(Dictionary info)

Add a custom property info to a property. The dictionary must contain: name:String, and optionally hint:int), hint_string:String.

Example:

```java
editor_settings.set("category/property_name", 0)

var property_info = {
    "name": "category/property_name",
    "type": TYPE_INT,
    "hint": PROPERTY_HINT_ENUM,
    "hint_string": "one,two,three"
}

editor_settings.add_property_info(property_info)
```

- void erase(String property)

Erase a given setting (pass full property path).

- StringArray get_favorite_dirs() const

Get the list of favorite directories for this project.

- String get_project_settings_path() const
Get the specific project settings path. Projects all have an unique sub-directory inside the settings path where project specific settings are saved.

- **`StringArray get_recent_dirs()`** const
  Get the list of recently visited folders in the file dialog for this project.

- **`String get_settings_path()`** const
  Get the global settings path for the engine. Inside this path you can find some standard paths such as:
  - settings/tmp - used for temporary storage of files
  - settings/templates - where export templates are located

- **`void set_favorite_dirs(StringArray dirs)`**
  Set the list of favorite directories for this project.

- **`void set_recent_dirs(StringArray dirs)`**
  Set the list of recently visited folders in the file dialog for this project.

### 12.89 EditorSpatialGizmo

**Inherits:** `SpatialGizmo < Reference < Object`

**Category:** Core

#### 12.89.1 Brief Description

Custom gizmo for editing Spatial objects.

#### 12.89.2 Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>add_collision_segments(Vector3Array segments)</code></td>
<td>Add collision segments</td>
</tr>
<tr>
<td><code>add_collision_triangles(TriangleMesh triangles)</code></td>
<td>Add collision triangles</td>
</tr>
<tr>
<td><code>add_handles(Vector3Array handles, bool billboard=false, bool secondary=false)</code></td>
<td>Add handles</td>
</tr>
<tr>
<td><code>add_lines(Vector3Array lines, Material material, bool billboard=false)</code></td>
<td>Add lines</td>
</tr>
<tr>
<td><code>add_mesh(Mesh mesh, bool billboard=false, RID skeleton=RID())</code></td>
<td>Add mesh</td>
</tr>
<tr>
<td><code>add_unscaled_billboard(Material material, float default_scale=1)</code></td>
<td>Add unscaled billboard</td>
</tr>
<tr>
<td><code>clear()</code></td>
<td>Clear</td>
</tr>
<tr>
<td><code>commit_handle(int index, Variant restore, bool cancel)</code> virtual</td>
<td>Commit handle</td>
</tr>
<tr>
<td><code>get_handle_name(int index)</code> virtual</td>
<td>Get handle name</td>
</tr>
<tr>
<td><code>get_handle_value(int index)</code> virtual</td>
<td>Get handle value</td>
</tr>
<tr>
<td><code>redraw()</code> virtual</td>
<td>Redraw</td>
</tr>
<tr>
<td><code>set_handle(int index, Camera camera, Vector2 point)</code></td>
<td>Set handle</td>
</tr>
<tr>
<td><code>set_spatial_node(Spatial node)</code></td>
<td>Set spatial node</td>
</tr>
</tbody>
</table>

#### 12.89.3 Description

Custom gizmo that is used for providing custom visualization and editing (handles) for 3D Spatial objects. These are created by `EditorPlugin.create_spatial_gizmo.`
12.89.4 Member Function Description

- **void add_collision_segments (Vector3Array segments)**
  Add collision triangles to the gizmo for picking. A TriangleMesh can be generated from a regular `Mesh` too. Call this function during `redraw`.

- **void add_collision_triangles (TriangleMesh triangles)**

- **void add_handles (Vector3Array handles, bool billboard=false, bool secondary=false)**
  Add a list of handles (points) which can be used to deform the object being edited. There are virtual functions which will be called upon editing of these handles. Call this function during `redraw`.

- **void add_lines (Vector3Array lines, Material material, bool billboard=false)**
  Add lines to the gizmo (as sets of 2 points), with a given material. The lines are used for visualizing the gizmo. Call this function during `redraw`.

- **void add_mesh (Mesh mesh, bool billboard=false, RID skeleton=RID())**
  Add a mesh to the gizmo, this is used for visualization. Call this function during `redraw`.

- **void add_unscaled_billboard (Material material, float default_scale=1)**
  Add an unscaled billboard for visualization. Call this function during `redraw`.

- **void clear ( )**

- **void commit_handle (int index, Variant restore, bool cancel) virtual**
  Commit a handle being edited (handles must have been previously added by `add_handles`). If the cancel parameter is true, an option to restore the edited value to the original is provided.

- **String get_handle_name (int index) virtual**
  Get the name of an edited handle (handles must have been previously added by `add_handles`). Handles can be named for reference to the user when editing.

- **Variant get_handle_value (int index) virtual**
  Get actual value of a handle. This value can be anything and used for eventually undoing the motion when calling `commit_handle`.

- **void redraw ( ) virtual**
  This function is called when the Spatial this gizmo refers to changes (the `Spatial.update_gizmo` is called).

- **void set_handle (int index, Camera camera, Vector2 point) virtual**
  This function is used when the user drags a gizmo handle (previously added with `add_handles`) in screen coordinates. The `Camera` is also provided so screen coordinates can be converted to raycasts.

- **void set_spatial_node (Spatial node)**
  Call this function once and upon creation of the gizmo, otherwise no other function will work. The argument is the node being edited by the gizmo.
12.90 Environment

Inherits: Resource < Reference < Object

Category: Core

12.90.1 Brief Description

12.90.2 Member Functions

<table>
<thead>
<tr>
<th>void</th>
<th>fx_get_param ( int param ) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>fx_set_param ( int param, Variant value )</td>
</tr>
<tr>
<td>int</td>
<td>get_background ( ) const</td>
</tr>
<tr>
<td>void</td>
<td>get_background_param ( int param ) const</td>
</tr>
<tr>
<td>bool</td>
<td>is_fx_enabled ( int effect ) const</td>
</tr>
<tr>
<td>void</td>
<td>set_background ( int bgmode )</td>
</tr>
<tr>
<td>void</td>
<td>set_background_param ( int param, Variant value )</td>
</tr>
<tr>
<td>void</td>
<td>set_enable_fx ( int effect, bool enabled )</td>
</tr>
</tbody>
</table>

12.90.3 Numeric Constants

• BG_KEEP = 0
• BG_DEFAULT_COLOR = 1
• BG_COLOR = 2
• BG_TEXTURE = 3
• BG_CUBEMAP = 4
• BG_CANVAS = 5
• BG_MAX = 6
• BG_PARAM_CANVAS_MAX_LAYER = 0
• BG_PARAM_COLOR = 1
• BG_PARAM_TEXTURE = 2
• BG_PARAM_CUBEMAP = 3
• BG_PARAM_ENERGY = 4
• BG_PARAM_GLOW = 6
• BG_PARAM_MAX = 7
• FX_AMBIENT_LIGHT = 0
• FX_FXAA = 1
• FX_GLOW = 2
• FX_DOF_BLUR = 3
• FX_HDR = 4
• FX_FOG = 5
• FX_BCS = 6
• FX_SRGB = 7
• FX_MAX = 8
• FX_BLUR_BLEND_MODE_ADDITIVE = 0
• FX_BLUR_BLEND_MODE_SCREEN = 1
• FX_BLUR_BLEND_MODE_SOFTLIGHT = 2
• FX_HDR_TONE_MAPPER_LINEAR = 0
• FX_HDR_TONE_MAPPER_LOG = 1
• FX_HDR_TONE_MAPPER_REINHARDT = 2
• FX_HDR_TONE_MAPPER_REINHARDT_AUTOWHITE = 3
• FX_PARAM_AMBIENT_LIGHT_COLOR = 0
• FX_PARAM_AMBIENT_LIGHT_ENERGY = 1
• FX_PARAM_GLOW_BLUR_PASSES = 2
• FX_PARAM_GLOW_BLUR_SCALE = 3
• FX_PARAM_GLOW_BLUR_STRENGTH = 4
• FX_PARAM_GLOW_BLUR_BLEND_MODE = 5
• FX_PARAM_GLOW_BLOOM = 6
• FX_PARAM_GLOW_BLOOM_TRESHOLD = 7
• FX_PARAM_DOF_BLUR_PASSES = 8
• FX_PARAM_DOF_BLUR_BEGIN = 9
• FX_PARAM_DOF_BLUR_RANGE = 10
• FX_PARAM_HDR_TONEMAPPER = 11
• FX_PARAM_HDR_EXPOSURE = 12
• FX_PARAM_HDR_WHITE = 13
• FX_PARAM_HDR_GLOW_TRESHOLD = 14
• FX_PARAM_HDR_GLOW_SCALE = 15
• FX_PARAM_HDR_MIN_LUMINANCE = 16
• FX_PARAM_HDR_MAX_LUMINANCE = 17
• FX_PARAM_HDR_EXPOSURE_ADJUST_SPEED = 18
• FX_PARAM_FOG_BEGIN = 19
• FX_PARAM_FOG_ATTENUATION = 22
• FX_PARAM_FOG_BEGIN_COLOR = 20
• FX_PARAM_FOG_END_COLOR = 21
• FX_PARAM_FOG_BG = 23
• FX_PARAM_BCS_BRIGHTNESS = 24
• FX_PARAM_BCS_CONTRAST = 25
• FX_PARAM_BCS_SATURATION = 26
• FX_PARAM_MAX = 27

12.90.4 Member Function Description

• void fx_get_param ( int param ) const
• void fx_set_param ( int param, Variant value )
• int get_background ( ) const
• void get_background_param ( int param ) const
• bool is_fx_enabled ( int effect ) const
• void set_background ( int bgmode )
• void set_background_param ( int param, Variant value )
• void set_enable_fx ( int effect, bool enabled )

12.91 EventPlayer

Inherits: Node < Object

Category: Core

12.91.1 Brief Description

Class for event stream playback.
12.91.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>float</td>
<td><code>get_channel_last_note_time (int channel )</code></td>
</tr>
<tr>
<td>float</td>
<td><code>get_channel_volume (int channel)</code></td>
</tr>
<tr>
<td>float</td>
<td><code>get_length()</code></td>
</tr>
<tr>
<td>int</td>
<td><code>get_loop_count()</code></td>
</tr>
<tr>
<td>float</td>
<td><code>get_pitch_scale()</code></td>
</tr>
<tr>
<td>float</td>
<td><code>get_pos()</code></td>
</tr>
<tr>
<td>EventStream</td>
<td><code>get_stream()</code></td>
</tr>
<tr>
<td>String</td>
<td><code>get_stream_name()</code></td>
</tr>
<tr>
<td>float</td>
<td><code>get_tempo_scale()</code></td>
</tr>
<tr>
<td>float</td>
<td><code>get_volume()</code></td>
</tr>
<tr>
<td>bool</td>
<td><code>has_autoplay()</code></td>
</tr>
<tr>
<td>bool</td>
<td><code>has_loop()</code></td>
</tr>
<tr>
<td>bool</td>
<td><code>is_paused()</code></td>
</tr>
<tr>
<td>bool</td>
<td><code>is_playing()</code></td>
</tr>
<tr>
<td>void</td>
<td><code>play()</code></td>
</tr>
<tr>
<td>void</td>
<td><code>seek_pos (float time )</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_autoplay (bool enabled)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_channel_volume (int channel, float channel_volume)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_loop (bool enabled)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_paused (bool paused)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_pitch_scale (float pitch_scale)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_stream (EventStream stream)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_tempo_scale (float tempo_scale)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_volume (float volume)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_volume_db (float db)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>stop()</code></td>
</tr>
</tbody>
</table>

12.91.3 Description

Class for event stream playback. Event streams are music expressed as a series of events (note on, note off, instrument change...), as opposed to audio streams, which are just audio data. Examples of event-based streams are MIDI files, or MOD music.

Currently, only MOD, S3M, IT, and XM music is supported.

12.91.4 Member Function Description

- `float get_channel_last_note_time (int channel)` const
  Return the time at which the last note of a given channel in the stream plays.

- `float get_channel_volume (int channel)` const
  Return the volume scale for an individual channel of the stream.

- `float get_length()` const
  Return the song length. May be in seconds, but depends on the stream type.

- `int get_loop_count()` const
  Return the number of times the playback has looped.
• float get_pitch_scale ( ) const
Return the pitch scale factor for this player.

• float get_pos ( ) const
Return the playback position. May be in seconds, but depends on the stream type.

• EventStream get_stream ( ) const
Return the currently assigned stream.

• String get_stream_name ( ) const
Return the name of the currently assigned stream. This is not the file name, but a field inside the file. If no stream is assigned, it returns "<No Stream>".

• float get_tempo_scale ( ) const
Return the tempo multiplier.

• float get_volume ( ) const
Return the playback volume for this player.

• float get_volume_db ( ) const
Return the playback volume for this player, in decibels.

• bool has_autoplay ( ) const
Return whether this player will start playing as soon as it enters the scene tree.

• bool has_loop ( ) const
Return whether this player will be restarted at the end.

• bool is_paused ( ) const
Return whether the playback is currently paused.

• bool is_playing ( ) const
Return whether this player is playing.

• void play ( )
Play the currently assigned stream.

• void seek_pos ( float time )
Set the playback position. May be in seconds, but depends on the stream type.

• void set_autoplay ( bool enabled )
Set whether this player will start playing as soon as it enters the scene tree.

• void set_channel_volume ( int channel, float channel_volume )
Set the volume scale for an individual channel of the stream, with the same value range as set_volume. The channel number depends on the stream format. For example, MIDIs range from 0 to 15, and MODs from 0 to 63.

Many stream formats are multichannel, so this allows to affect only a part of the music.

• void set_loop ( bool enabled )
Set whether the stream will be restarted at the end.

• void set_paused ( bool paused )
Pause stream playback.
• void set_pitch_scale (float pitch_scale)
Set the pitch multiplier for all sounds coming from this stream. A value of 2.0 shifts all pitches one octave up, and a value of 0.5 shifts pitches one octave down.

• void set_stream (EventStream stream)
Set the EventStream this player will play.

• void set_tempo_scale (float tempo_scale)
Set the tempo multiplier. This allows to slow down or speed up the music, without affecting its pitch.

• void set_volume (float volume)
Set the playback volume for this player. This is a float between 0.0 (silent) and 1.0 (full volume). Values over 1.0 may amplify sound even more, but may introduce distortion. Negative values may just invert the output waveform, which produces no audible difference.

The effect of these special values ultimately depends on the low-level implementation of the file format being played.

• void set_volume_db (float db)
Set the playback volume for this player, in decibels. This is a float between -80.0 (silent) and 0.0 (full volume). Values under -79.0 get truncated to -80, but values over 0.0 do not, so the warnings for over amplifying (see set_volume) still apply.

• void stop ()
Stop playing.

12.92 EventStream

Inherits: Resource < Reference < Object

Inherited By: EventStreamChibi

Category: Core

12.92.1 Brief Description

Base class for all event-based stream drivers.

12.92.2 Description

Base class for all event-based stream drivers. Event streams are music expressed as a series of events (note on, note off, instrument change...), as opposed to audio streams, which are just audio data. Examples of event-based streams are MIDI files, of MOD music.

This class exposes no methods.

12.93 EventStreamChibi

Inherits: EventStream < Resource < Reference < Object

Category: Core
12.93.1 Brief Description

Driver for MOD playback.

12.93.2 Description

This driver plays MOD music. MOD music, as all event-based streams, is a music format defined by note events occurring at defined moments, instead of a stream of audio samples.

Currently, this driver supports the MOD, S3M, IT, and XM formats.

This class exposes no methods.

This class can return its playback position in seconds, but does not allow to set it, failing with only a console warning.

This class can not return its song length, returning 1.0 when queried.

This class does not limit its volume settings, allowing for overflow/distortion and wave inversion.

12.94 File

Inherits: Reference < Object

Category: Core

12.94.1 Brief Description

Type to handle file reading and writing operations.

12.94.2 Member Functions

<table>
<thead>
<tr>
<th>void</th>
<th>close ( )</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool</td>
<td>eof_reached ( ) const</td>
</tr>
<tr>
<td>bool</td>
<td>file_exists ( String path ) const</td>
</tr>
<tr>
<td>int</td>
<td>get_16 ( ) const</td>
</tr>
<tr>
<td>int</td>
<td>get_32 ( ) const</td>
</tr>
<tr>
<td>int</td>
<td>get_64 ( ) const</td>
</tr>
<tr>
<td>int</td>
<td>get_8 ( ) const</td>
</tr>
<tr>
<td>String</td>
<td>get_as_text ( ) const</td>
</tr>
<tr>
<td>RawArray</td>
<td>get_buffer ( int len ) const</td>
</tr>
<tr>
<td>StringArray</td>
<td>get_csv_line ( String delim=&quot;&quot;, ) const</td>
</tr>
<tr>
<td>float</td>
<td>get_double ( ) const</td>
</tr>
<tr>
<td>bool</td>
<td>get_endian_swap ( )</td>
</tr>
<tr>
<td>Error</td>
<td>get_error ( ) const</td>
</tr>
<tr>
<td>float</td>
<td>get_float ( ) const</td>
</tr>
<tr>
<td>int</td>
<td>get_len ( ) const</td>
</tr>
<tr>
<td>String</td>
<td>get_line ( ) const</td>
</tr>
<tr>
<td>String</td>
<td>get_md5 ( String path ) const</td>
</tr>
<tr>
<td>int</td>
<td>get_modified_time ( String file ) const</td>
</tr>
<tr>
<td>String</td>
<td>get_pascal_string ( )</td>
</tr>
</tbody>
</table>
12.94.3 Numeric Constants

- **READ** = 1 — Open the file for reading.
- **WRITE** = 2 — Open the file for writing. Create it if the file not exists and truncate if it exists.
- **READ_WRITE** = 3 — Open the file for reading and writing, without truncating the file.
- **WRITE_READ** = 7 — Open the file for reading and writing. Create it if the file not exists and truncate if it exists.

12.94.4 Description

File type. This is used to permanently store data into the user device’s file system and to read from it. This can be used to store game save data or player configuration files, for example.

Here’s a sample on how to write and read from a file:

```godot
func save(content):
    var file = File.new()
    file.open("user://save_game.dat", file.WRITE)
    file.store_string(content)
    file.close()

func load():
    var file = File.new()
    file.open("user://save_game.dat", file.READ)
    var content = file.get_as_text()
    file.close()
    return content
```
12.94.5 Member Function Description

- **void** close ()
  Close the currently opened file.

- **bool** eof_reached () const
  Return whether the file cursor reached the end of the file.

- **bool** file_exists (String path) const
  Get whether or not the file in the specified path exists.

- **int** get_16 () const
  Get the next 16 bits from the file as an integer.

- **int** get_32 () const
  Get the next 32 bits from the file as an integer.

- **int** get_64 () const
  Get the next 64 bits from the file as an integer.

- **int** get_8 () const
  Get the next 8 bits from the file as an integer.

- **String** get_as_text () const
  Get the whole file as a String.

- **RawArray** get_buffer (int len) const
  Get next len bytes of the file as a RawArray.

- **StringArray** get_csv_line (String delim = "", ) const
  Get the next value of the file in CSV (Comma Separated Values) format. You can pass a different delimiter to use other than the default "," (comma).

- **float** get_double () const
  Get the next 64 bits from the file as a floating point number.

- **bool** get_endian_swap ()
  Get whether endian swap is enabled for this file.

- **Error** get_error () const
  Get the last error that happened when trying to perform operations. Compare with the ERR_FILE\_\* constants from @Global Scope.

- **float** get_float () const
  Get the next 32 bits from the file as a floating point number.

- **int** get_len () const
  Return the size of the file in bytes.

- **String** get_line () const
Get the next line of the file as a *String*.

- *String* get_md5 (*String* path ) const
  
  Return a md5 String representing the file at the given path or an empty *String* on failure.

- *int* get_modified_time (*String* file ) const

- *String* get_pascal_string ()
  
  Get a *String* saved in Pascal format from the file.

- *int* get_pos () const
  
  Return the file cursor position.

- *float* get_real () const
  
  Get the next bits from the file as a floating point number.

- *String* get_sha256 (*String* path ) const
  
  Return a sha256 String representing the file at the given path or an empty *String* on failure.

- void get_var () const
  
  Get the next Variant value from the file.

- *bool* is_open () const
  
  Return whether the file is currently opened.

- *int* open (*String* path, *int* flags )
  
  Open the file for writing or reading, depending on the flags.

- *int* open_encrypted (*String* path, *int* mode_flags, *RawArray* key )
  
  Open an encrypted file in write or read mode. You need to pass a binary key to encrypt/decrypt it.

- *int* open_encrypted_with_pass (*String* path, *int* mode_flags, *String* pass )
  
  Open an encrypted file in write or read mode. You need to pass a password to encrypt/decrypt it.

- void seek (*int* pos )
  
  Change the file reading/writing cursor to the specified position (in bytes from the beginning of the file).

- void seek_end (*int* pos=0 )
  
  Change the file reading/writing cursor to the specified position (in bytes from the end of the file). Note that this is an offset, so you should use negative numbers or the cursor will be at the end of the file.

- void set_endian_swap (*bool* enable )
  
  Set whether to swap the endianess of the file. Enable this if you’re dealing with files written in big endian machines. Note that this is about the file format, not CPU type. This is always reseted to false whenever you open the file.

- void store_16 (*int* value )
  
  Store an integer as 16 bits in the file.

- void store_32 (*int* value )
  
  Store an integer as 32 bits in the file.

- void store_64 (*int* value )
  
  Store an integer as 64 bits in the file.
• void store_8 ( int value )
  Store an integer as 8 bits in the file.
• void store_buffer ( RawArray buffer )
  Store the given array of bytes in the file.
• void store_double ( float value )
  Store a floating point number as 64 bits in the file.
• void store_float ( float value )
  Store a floating point number as 32 bits in the file.
• void store_line ( String line )
  Store the given String as a line in the file.
• void store_pascal_string ( String string )
  Store the given String as a line in the file in Pascal format (i.e. also store the length of the string).
• void store_real ( float value )
  Store a floating point number in the file.
• void store_string ( String string )
  Store the given String in the file.
• void store_var ( Variant value )
  Store any Variant value in the file.

12.95 FileDialog

Inherits: ConfirmationDialog < AcceptDialog < WindowDialog < Popup < Control < CanvasItem < Node < Object

Category: Core

12.95.1 Brief Description

Dialog for selecting files or directories in the filesystem.
12.95.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>add_filter(String filter)</td>
</tr>
<tr>
<td>void</td>
<td>clear_filters()</td>
</tr>
<tr>
<td>int</td>
<td>get_access() const</td>
</tr>
<tr>
<td>String</td>
<td>get_current_dir() const</td>
</tr>
<tr>
<td>String</td>
<td>get_current_file() const</td>
</tr>
<tr>
<td>String</td>
<td>get_current_path() const</td>
</tr>
<tr>
<td>StringArray</td>
<td>get_filters() const</td>
</tr>
<tr>
<td>int</td>
<td>get_mode() const</td>
</tr>
<tr>
<td>VBoxContainer</td>
<td>get_vbox()</td>
</tr>
<tr>
<td>void</td>
<td>invalidate()</td>
</tr>
<tr>
<td>bool</td>
<td>is_showing_hidden_files() const</td>
</tr>
<tr>
<td>void</td>
<td>set_access(int access)</td>
</tr>
<tr>
<td>void</td>
<td>set_current_dir(String dir)</td>
</tr>
<tr>
<td>void</td>
<td>set_current_file(String file)</td>
</tr>
<tr>
<td>void</td>
<td>set_current_path(String path)</td>
</tr>
<tr>
<td>void</td>
<td>set_filters(StringArray filters)</td>
</tr>
<tr>
<td>void</td>
<td>set_mode(int mode)</td>
</tr>
<tr>
<td>void</td>
<td>set_show_hidden_files(bool show)</td>
</tr>
</tbody>
</table>

12.95.3 Signals

- **dir_selected(String dir)**
  Event emitted when the user selects a directory.

- **file_selected(String path)**
  Event emitted when the user selects a file (double clicks it or presses the OK button).

- **files_selected(StringArray paths)**
  Event emitted when the user selects multiple files.

12.95.4 Numeric Constants

- **MODE_OPEN_FILE = 0** — The dialog allows the selection of one, and only one file.
- **MODE_OPEN_FILES = 1** — The dialog allows the selection of multiple files.
- **MODE_OPEN_DIR = 2** — The dialog functions as a folder selector, disallowing the selection of any file.
- **MODE_OPEN_ANY = 3** — The dialog allows the selection of a file or a directory.
- **MODE_SAVE_FILE = 4** — The dialog will warn when a file exists.
- **ACCESS_RESOURCES = 0** — The dialog allows the selection of file and directory.
- **ACCESS_USERDATA = 1** — The dialog allows access files under Resource path(res://) .
- **ACCESS_FILESYSTEM = 2** — The dialog allows access files in whole file system.

12.95.5 Description

FileDialog is a preset dialog used to choose files and directories in the filesystem. It supports filter masks.
12.95.6 Member Function Description

- void **add_filter** ( String filter )
  Add a custom filter. Filter format is: “mask ; description”, example (C++): dialog->add_filter("*.png ; PNG Images");

- void **clear_filters** ( )
  Clear all the added filters in the dialog.

- int **get_access** ( ) const
  Return the file access permission of the dialog.

- String **get_current_dir** ( ) const
  Get the current working directory of the file dialog.

- String **get_current_file** ( ) const
  Get the current selected file of the file dialog (empty if none).

- String **get_current_path** ( ) const
  Get the current selected path (directory and file) of the file dialog (empty if none).

- StringArray **get_filters** ( ) const
  Get the file dialog mode from the MODE_* enum.

- VBoxContainer **get_vbox** ( )
  Return the vertical box container of the dialog, custom controls can be added to it.

- void **invalidate** ( )
  Invalidate and update the current dialog content list.

- bool **is_showing_hidden_files** ( ) const
  Return true if the dialog allows show hidden files.

- void **set_access** ( int access )
  Set the file access permission of the dialog (Must be one of ACCESS_RESOURCES, ACCESS_USERDATA or ACCESS_FILESYSTEM).

- void **set_current_dir** ( String dir )
  Set the current working directory of the file dialog.

- void **set_current_file** ( String file )
  Set the current selected file name of the file dialog.

- void **set_current_path** ( String path )
  Set the current selected file path of the file dialog.

- void **set_filters** ( StringArray filters )

- void **set_mode** ( int mode )
  Set the file dialog mode from the MODE_* enum.

- void **set_show_hidden_files** ( bool show )

Set the dialog should show hidden files.

12.96 FixedMaterial

Inherits: Material < Resource < Reference < Object

Category: Core

12.96.1 Brief Description

Simple Material with a fixed parameter set.

12.96.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool</td>
<td><code>get_fixed_flag(int flag)</code> const</td>
</tr>
<tr>
<td>int</td>
<td><code>get_light_shader()</code> const</td>
</tr>
<tr>
<td>void</td>
<td><code>get_parameter(int param)</code> const</td>
</tr>
<tr>
<td>float</td>
<td><code>get_point_size()</code> const</td>
</tr>
<tr>
<td>int</td>
<td><code>get_texcoord_mode(int param)</code> const</td>
</tr>
<tr>
<td>Texture</td>
<td><code>get_texture(int param)</code> const</td>
</tr>
<tr>
<td>Transform</td>
<td><code>get_uv_transform()</code> const</td>
</tr>
<tr>
<td>void</td>
<td><code>set_fixed_flag(int flag, bool value)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_light_shader(int shader)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_parameter(int param, Variant value)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_point_size(float size)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_texcoord_mode(int param, int mode)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_texture(int param, Texture texture)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_uv_transform(Transform transform)</code></td>
</tr>
</tbody>
</table>

12.96.3 Numeric Constants

- `PARAM_DIFFUSE = 0` — Diffuse Lighting (light scattered from surface).
- `PARAM_DETAIL = 1` — Detail Layer for diffuse lighting.
- `PARAM_SPECULAR = 2` — Specular Lighting (light reflected from the surface).
- `PARAM_EMISSION = 3` — Emission Lighting (light emitted from the surface).
- `PARAM_SPECULAR_EXP = 4` — Specular Exponent (size of the specular dot).
- `PARAM_GLOW = 5` — Glow (Visible emitted scattered light).
- `PARAM_NORMAL = 6` — Normal Map (irregularity map).
- `PARAM_SHADE_PARAM = 7`
- `PARAM_MAX = 8` — Maximum amount of parameters.
- `TEXCOORD_UV = 0` — Read texture coordinates from the UV array.
- `TEXCOORD_UV_TRANSFORM = 1` — Read texture coordinates from the UV array and transform them by `uv_xform`.
- `TEXCOORD_UV2 = 2` — Read texture coordinates from the UV2 array.
12.96.4 Description

FixedMaterial is a simple type of material Resource, which contains a fixed amount of parameters. It is the only type of material supported in fixed-pipeline devices and APIs. It is also an often a better alternative to ShaderMaterial for most simple use cases.

12.96.5 Member Function Description

- bool get_fixed_flag (int flag) const
- int get_light_shader () const
- void get_parameter (int param) const

Return a parameter, parameters are defined in the PARAM_* enum. The type of each parameter may change, so it’s best to check the enum.

- float get_point_size () const
- int get_texcoord_mode (int param) const

Return the texture coordinate mode. Each texture param (from the PARAM_* enum) has one. It defines how the textures are mapped to the object.

- Texture get_texture (int param) const

Return a texture. Textures change parameters per texel and are mapped to the model depending on the texcoord mode (see set_texcoord_mode).

- Transform get_uv_transform () const

Returns the special transform used to post-transform UV coordinates of the uv_xform texcoord mode: TEXCOORD_UV_TRANSFORM.

- void set_fixed_flag (int flag, bool value )
- void set_light_shader (int shader )
- void set_parameter (int param, Variant value )

Set a parameter, parameters are defined in the PARAM_* enum. The type of each parameter may change, so it’s best to check the enum.

- void set_point_size (float size )
- void set_texcoord_mode (int param, int mode )
Set the texture coordinate mode. Each texture param (from the PARAM_* enum) has one. It defines how the textures are mapped to the object.

- **void set_texture ( int param, Texture texture )**

Set a texture. Textures change parameters per texel and are mapped to the model depending on the texcoord mode (see set_texcoord_mode).

- **void set_uv_transform ( Transform transform )**

Sets a special transform used to post-transform UV coordinates of the uv_xform texcoord mode: TEXCOORD_UV_TRANSFORM.

### 12.97 float

**Category:** Built-In Types

#### 12.97.1 Brief Description

Float built-in type

#### 12.97.2 Member Functions

<table>
<thead>
<tr>
<th>float</th>
<th>float ( bool from )</th>
</tr>
</thead>
<tbody>
<tr>
<td>float</td>
<td>float ( int from )</td>
</tr>
<tr>
<td>float</td>
<td>float ( String from )</td>
</tr>
</tbody>
</table>

#### 12.97.3 Description

Float built-in type.

#### 12.97.4 Member Function Description

- **float float ( bool from )**

Cast a **bool** value to a floating point value, **float(true)** will be equals to 1.0 and **float(false)** will be equals to 0.0.

- **float float ( int from )**

Cast an **int** value to a floating point value, **float(1)** will be equals to 1.0.

- **float float ( String from )**

Cast a **String** value to a floating point value. This method accepts float value strings like `'1.23'` and exponential notation strings for its parameter so calling `'float('1e3')` will return 1000.0 and calling `'float('1e-3')` will return -0.001.
12.98 Font

Inherits: Resource < Reference < Object

Inherited By: DynamicFont, BitmapFont

Category: Core

12.98.1 Brief Description

Internationalized font and text drawing support.

12.98.2 Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Declaration</th>
</tr>
</thead>
<tbody>
<tr>
<td>void draw ( RID canvas_item, Vector2 pos, String string, Color modulate=Color(1,1,1,1), int clip_w=-1 ) const</td>
<td></td>
</tr>
<tr>
<td>float draw_char ( RID canvas_item, Vector2 pos, int char, int next=-1, Color modulate=Color(1,1,1,1) ) const</td>
<td></td>
</tr>
<tr>
<td>float get_ascent () const</td>
<td></td>
</tr>
<tr>
<td>float get_descent () const</td>
<td></td>
</tr>
<tr>
<td>float get_height () const</td>
<td></td>
</tr>
<tr>
<td>Vector2 get_string_size ( String string ) const</td>
<td></td>
</tr>
<tr>
<td>bool is_distance_field_hint () const</td>
<td></td>
</tr>
<tr>
<td>void update_changes ()</td>
<td></td>
</tr>
</tbody>
</table>

12.98.3 Description

Font contains an unicode compatible character set, as well as the ability to draw it with variable width, ascent, descent and kerning. For creating fonts from TTF files (or other font formats), see the editor support for fonts. TODO check wikipedia for graph of ascent/baseline/descent/height/etc.

12.98.4 Member Function Description

- void draw ( RID canvas_item, Vector2 pos, String string, Color modulate=Color(1,1,1,1), int clip_w=-1 ) const
  Draw “string” into a canvas item using the font at a given “pos” position, with “modulate” color, and optionally clipping the width. “pos” specifies the baseline, not the top. To draw from the top, ascent must be added to the Y axis.

- float draw_char ( RID canvas_item, Vector2 pos, int char, int next=-1, Color modulate=Color(1,1,1,1) ) const
  Draw character “char” into a canvas item using the font at a given “pos” position, with “modulate” color, and optionally kerning if “next” is passed. clipping the width. “pos” specifies the baseline, not the top. To draw from the top, ascent must be added to the Y axis. The width used by the character is returned, making this function useful for drawing strings character by character.

- float get_ascent () const
  Return the font ascent (number of pixels above the baseline).

- float get_descent () const
  Return the font descent (number of pixels below the baseline).
• `float get_height ( ) const`
Return the total font height (ascent plus descent) in pixels.

• `Vector2 get_string_size ( String string ) const`
Return the size of a string, taking kerning and advance into account.

• `bool is_distance_field_hint ( ) const`

• `void update_changes ( )`
After editing a font (changing size, ascent, char rects, etc.). Call this function to propagate changes to controls that might use it.

12.99 FuncRef

Inherits: `Reference < Object`

Category: Core

12.99.1 Brief Description

Reference to a function in an object.

12.99.2 Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void call_func ( Variant arg0=NULL, Variant arg1=NULL, Variant arg2=NULL, Variant arg3=NULL, Variant arg4=NULL, Variant arg5=NULL, Variant arg6=NULL, Variant arg7=NULL, Variant arg8=NULL, Variant arg9=NULL )</code></td>
<td>Call the referenced function with the given arguments. The argument count must correspond to the required number of arguments in the function. Returns the return value of the function call.</td>
</tr>
<tr>
<td><code>void set_function ( String name )</code></td>
<td>Set the name of the function to call on the object, without parentheses or any parameters.</td>
</tr>
</tbody>
</table>

12.99.3 Description

In GDScript, functions are not first-class objects. This means it is impossible to store them directly as variables, return them from another function, or pass them as arguments.

However, by creating a `FuncRef` using the `@GDScript.funcref` function, a reference to a function in a given object can be created, passed around and called.

12.99.4 Member Function Description

• `void call_func ( Variant arg0=NULL, Variant arg1=NULL, Variant arg2=NULL, Variant arg3=NULL, Variant arg4=NULL, Variant arg5=NULL, Variant arg6=NULL, Variant arg7=NULL, Variant arg8=NULL, Variant arg9=NULL )`

Call the referenced function with the given arguments. The argument count must correspond to the required number of arguments in the function. Returns the return value of the function call.

• `void set_function ( String name )`
Set the name of the function to call on the object, without parentheses or any parameters.
• void setInstance (Object instance)

Set the object on which to call the referenced function. This object must be of a type actually inheriting from Object, not a built-in type such as int, Vector2 or Dictionary.

12.100 GDFunctionState

Inherits: Reference < Object

Category: Core

12.100.1 Brief Description

State of a function call after yielding.

12.100.2 Member Functions

<table>
<thead>
<tr>
<th>bool</th>
<th>is_valid ( bool extended_check=false ) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variant</td>
<td>resume ( Variant arg=NULL )</td>
</tr>
</tbody>
</table>

12.100.3 Description

Calling @GDScript.yield within a function will cause that function to yield and return its current state as an object of this type. The yielded function call can then be resumed later by calling resume on this state object.

12.100.4 Member Function Description

• bool is_valid ( bool extended_check=false ) const

Check whether the function call may be resumed. This is not the case if the function state was already resumed.

If extended_check is enabled, it also checks if the associated script and object still exist. The extended check is done in debug mode as part of GDFunctionState.resume, but you can use this if you know you may be trying to resume without knowing for sure the object and/or script have survived up to that point.

• Variant resume ( Variant arg=NULL )

Resume execution of the yielded function call.

If handed an argument, return the argument from the @GDScript.yield call in the yielded function call. You can pass e.g. an Array to hand multiple arguments.

This function returns what the resumed function call returns, possibly another function state if yielded again.

12.101 GDNativeClass

Inherits: Reference < Object

Category: Core
12.101.1 Brief Description

12.101.2 Member Functions

| void | new () |

12.101.3 Member Function Description

- void new ()

12.102 GDScript

Inherits: Script < Resource < Reference < Object

Category: Core

12.102.1 Brief Description

12.102.2 Member Functions

<table>
<thead>
<tr>
<th>RawArray</th>
<th>get_as_byte_code () const</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>new ()</td>
</tr>
</tbody>
</table>

12.102.3 Member Function Description

- RawArray get_as_byte_code () const
- void new ()

12.103 Generic6DOFJoint

Inherits: Joint < Spatial < Node < Object

Category: Core
12.103.1 Brief Description

12.103.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool</td>
<td>get_flag_x (int flag)</td>
<td>const</td>
<td></td>
</tr>
<tr>
<td>bool</td>
<td>get_flag_y (int flag)</td>
<td>const</td>
<td></td>
</tr>
<tr>
<td>bool</td>
<td>get_flag_z (int flag)</td>
<td>const</td>
<td></td>
</tr>
<tr>
<td>float</td>
<td>get_param_x (int param)</td>
<td>const</td>
<td></td>
</tr>
<tr>
<td>float</td>
<td>get_param_y (int param)</td>
<td>const</td>
<td></td>
</tr>
<tr>
<td>float</td>
<td>get_param_z (int param)</td>
<td>const</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>set_flag_x (int flag, bool value)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>set_flag_y (int flag, bool value)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>set_flag_z (int flag, bool value)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>set_param_x (int param, float value)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>set_param_y (int param, float value)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>set_param_z (int param, float value)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

12.103.3 Numeric Constants

- PARAM_LINEAR_LOWER_LIMIT = 0
- PARAM_LINEAR_UPPER_LIMIT = 1
- PARAM_LINEAR_LIMIT_SOFTNESS = 2
- PARAM_LINEAR_RESTITUTION = 3
- PARAM_LINEAR_DAMPING = 4
- PARAM_ANGULAR_LOWER_LIMIT = 5
- PARAM_ANGULAR_UPPER_LIMIT = 6
- PARAM_ANGULAR_LIMIT_SOFTNESS = 7
- PARAM_ANGULAR_DAMPING = 8
- PARAM_ANGULAR_RESTITUTION = 9
- PARAM_ANGULAR_FORCE_LIMIT = 10
- PARAM_ANGULAR_ERP = 11
- PARAM_ANGULAR_MOTOR_TARGET_VELOCITY = 12
- PARAM_ANGULAR_MOTOR_FORCE_LIMIT = 13
- PARAM_MAX = 14
- FLAG_ENABLE_LINEAR_LIMIT = 0
- FLAG_ENABLE_ANGULAR_LIMIT = 1
- FLAG_ENABLE_MOTOR = 2
- FLAG_MAX = 3
12.103.4 Member Function Description

- `bool get_flag_x(int flag) const`
- `bool get_flag_y(int flag) const`
- `bool get_flag_z(int flag) const`
- `float get_param_x(int param) const`
- `float get_param_y(int param) const`
- `float get_param_z(int param) const`
- `void set_flag_x(int flag, bool value)`
- `void set_flag_y(int flag, bool value)`
- `void set_flag_z(int flag, bool value)`
- `void set_param_x(int param, float value)`
- `void set_param_y(int param, float value)`
- `void set_param_z(int param, float value)`

12.104 Geometry

Inherits: Object

Category: Core
12.104.1 Brief Description

12.104.2 Member Functions

<table>
<thead>
<tr>
<th>Array</th>
<th>build_box_planes ( Vector3 extents )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>build_capsule_planes ( float radius, float height, int sides, int lats, int axis=2 )</td>
</tr>
<tr>
<td>Array</td>
<td>build_cylinder_planes ( float radius, float height, int sides, int axis=2 )</td>
</tr>
<tr>
<td>Vector3</td>
<td>get_closest_point_to_segment ( Vector3 point, Vector3 s1, Vector3 s2 )</td>
</tr>
<tr>
<td>Vector2</td>
<td>get_closest_point_to_segment_2d ( Vector2 point, Vector2 s1, Vector2 s2 )</td>
</tr>
<tr>
<td>Vector3</td>
<td>get_closest_point_to_segment_uncapped ( Vector3 point, Vector3 s1, Vector3 s2 )</td>
</tr>
<tr>
<td>Vector2</td>
<td>get_closest_point_to_segment_uncapped_2d ( Vector2 point, Vector2 s1, Vector2 s2 )</td>
</tr>
<tr>
<td>Vector3Array</td>
<td>get_closest_points_between_segments ( Vector3 p1, Vector3 p2, Vector3 q1, Vector3 q2 )</td>
</tr>
<tr>
<td>Vector2Array</td>
<td>get_closest_points_between_segments_2d ( Vector2 p1, Vector2 q1, Vector2 p2, Vector2 q2 )</td>
</tr>
<tr>
<td>Int</td>
<td>get_uv84_normal_bit ( Vector3 normal )</td>
</tr>
<tr>
<td>Dictionary</td>
<td>make_atlas ( Vector2Array sizes )</td>
</tr>
<tr>
<td>bool</td>
<td>point_is_inside_triangle ( Vector2 point, Vector2 a, Vector2 b, Vector2 c ) const</td>
</tr>
<tr>
<td>void</td>
<td>ray_intersects_triangle ( Vector3 from, Vector3 dir, Vector3 a, Vector3 b, Vector3 c )</td>
</tr>
<tr>
<td>float</td>
<td>segment_intersects_circle ( Vector2 segment_from, Vector2 segment_to, Vector2 circle_pos, float circle_radius )</td>
</tr>
<tr>
<td>Vector3Array</td>
<td>segment_intersects_convex ( Vector3 from, Vector3 to, Array planes )</td>
</tr>
<tr>
<td>Vector3Array</td>
<td>segment_intersects_cylinder ( Vector3 from, Vector3 to, float height, float radius )</td>
</tr>
<tr>
<td>void</td>
<td>segment_intersects_segment_2d ( Vector2 from_a, Vector2 to_a, Vector2 from_b, Vector2 to_b )</td>
</tr>
<tr>
<td>Vector3Array</td>
<td>segment_intersects_sphere ( Vector3 from, Vector3 to, Vector3 spos, float sradius )</td>
</tr>
<tr>
<td>void</td>
<td>segment_intersects_sphere ( Vector3 from, Vector3 to, Vector3 spos, float sradius )</td>
</tr>
<tr>
<td>IntArray</td>
<td>triangulate_polygon ( Vector2Array polygon )</td>
</tr>
</tbody>
</table>

12.104.3 Member Function Description

- Array build_box_planes ( Vector3 extents )
- Array build_capsule_planes ( float radius, float height, int sides, int lats, int axis=2 )
- Array build_cylinder_planes ( float radius, float height, int sides, int axis=2 )
- Vector3 get_closest_point_to_segment ( Vector3 point, Vector3 s1, Vector3 s2 )
- Vector2 get_closest_point_to_segment_2d ( Vector2 point, Vector2 s1, Vector2 s2 )
- Vector3 get_closest_point_to_segment_uncapped ( Vector3 point, Vector3 s1, Vector3 s2 )
- Vector2 get_closest_point_to_segment_uncapped_2d ( Vector2 point, Vector2 s1, Vector2 s2 )
- Vector3Array get_closest_points_between_segments ( Vector3 p1, Vector3 p2, Vector3 q1, Vector3 q2 )
- Vector2Array get_closest_points_between_segments_2d ( Vector2 p1, Vector2 q1, Vector2 p2, Vector2 q2 )
- int get_uv84_normal_bit ( Vector3 normal )
- Dictionary make_atlas ( Vector2Array sizes )
- bool point_is_inside_triangle ( Vector2 point, Vector2 a, Vector2 b, Vector2 c ) const
- void ray_intersects_triangle ( Vector3 from, Vector3 dir, Vector3 a, Vector3 b, Vector3 c )
- float segment_intersects_circle ( Vector2 segment_from, Vector2 segment_to, Vector2 circle_pos, float circle_radius )
- Vector3Array segment_intersects_convex ( Vector3 from, Vector3 to, Array planes )
- Vector3Array segment_intersects_cylinder ( Vector3 from, Vector3 to, float height, float radius )
- void segment_intersects_segment_2d ( Vector2 from_a, Vector2 to_a, Vector2 from_b, Vector2 to_b )
- Vector3Array segment_intersects_sphere ( Vector3 from, Vector3 to, Vector3 spos, float sradius )
- void segment_intersects_triangle ( Vector3 from, Vector3 to, Vector3 a, Vector3 b, Vector3 c )
- IntArray triangulate_polygon ( Vector2Array polygon )

12.105 GeometryInstance

Inherits: VisualInstance < Spatial < Node < Object

Inherited By: MultiMeshInstance, SpriteBase3D, MeshInstance, Particles, Quad, TestCube, ImmediateGeometry

Category: Core

12.105.1 Brief Description

Base node for geometry based visual instances.

12.105.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>get_baked_light_texture_id () const</td>
<td>get baked light texture ID</td>
</tr>
<tr>
<td>int</td>
<td>get_cast_shadows_setting () const</td>
<td>get cast shadows setting</td>
</tr>
<tr>
<td>float</td>
<td>get_draw_range_begin () const</td>
<td>get draw range begin</td>
</tr>
<tr>
<td>float</td>
<td>get_draw_range_end () const</td>
<td>get draw range end</td>
</tr>
<tr>
<td>float</td>
<td>get_extra_cull_margin () const</td>
<td>get extra cull margin</td>
</tr>
<tr>
<td>bool</td>
<td>get_flag ( int flag ) const</td>
<td>get flag</td>
</tr>
<tr>
<td>Object</td>
<td>get_material_override () const</td>
<td>get material override</td>
</tr>
<tr>
<td>void</td>
<td>set_baked_light_texture_id ( int id )</td>
<td>set baked light texture ID</td>
</tr>
<tr>
<td>void</td>
<td>set_cast_shadows_setting ( int shadow_casting_setting )</td>
<td>set cast shadows setting</td>
</tr>
<tr>
<td>void</td>
<td>set_draw_range_begin ( float mode )</td>
<td>set draw range begin</td>
</tr>
<tr>
<td>void</td>
<td>set_draw_range_end ( float mode )</td>
<td>set draw range end</td>
</tr>
<tr>
<td>void</td>
<td>set_extra_cull_margin ( float margin )</td>
<td>set extra cull margin</td>
</tr>
<tr>
<td>void</td>
<td>set_flag ( int flag, bool value )</td>
<td>set flag</td>
</tr>
<tr>
<td>void</td>
<td>set_material_override ( Object material )</td>
<td>set material override</td>
</tr>
</tbody>
</table>

12.105.3 Numeric Constants

- FLAG_VISIBLE = 0
- FLAG_CAST_SHADOW = 3
- FLAG_RECEIVE_SHADOWS = 4
- FLAG_BILLBOARD = 1
- FLAG_BILLBOARD_FIX_Y = 2
• FLAG_DEPH_SCALE = 5
• FLAG_VISIBLE_IN_ALL_ROOMS = 6
• FLAG_MAX = 8
• SHADOW_CASTING_SETTING_OFF = 0
• SHADOW_CASTING_SETTING_ON = 1
• SHADOW_CASTING_SETTING_DOUBLE_SIDED = 2
• SHADOW_CASTING_SETTING_SHADOWS_ONLY = 3

### 12.105.4 Description

Base node for geometry based visual instances. Shares some common functionality like visibility and custom materials.

### 12.105.5 Member Function Description

- `int get_baked_light_texture_id ( ) const`
- `int get_cast_shadows_setting ( ) const`
- `float get_draw_range_begin ( ) const`
- `float get_draw_range_end ( ) const`
- `float get_extra_cull_margin ( ) const`
- `bool get_flag ( int flag ) const`
- `Object get_material_override ( ) const`

Return the material override for the whole geometry.

- `void set_baked_light_texture_id ( int id )`
- `void set_cast_shadows_setting ( int shadow_casting_setting )`
- `void set_draw_range_begin ( float mode )`
- `void set_draw_range_end ( float mode )`
- `void set_extra_cull_margin ( float margin )`
- `void set_flag ( int flag, bool value )`
- `void set_material_override ( Object material )`

Set the material override for the whole geometry.

### 12.106 Globals

Inherits: `Object`

Category: Core
12.106.1 Brief Description

Contains global variables accessible from everywhere.

12.106.2 Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void add_property_info(Dictionary hint)</td>
<td>Add a custom property info to a property. The dictionary must contain: name:String, and optionally hint:int), hint_string:String. Example:</td>
</tr>
<tr>
<td>void clear(String name)</td>
<td>Clear the whole configuration (not recommended, may break things).</td>
</tr>
<tr>
<td>int get_order(String name) const</td>
<td>Return the order of a configuration value (influences when saved to the config file).</td>
</tr>
<tr>
<td>Object get_singleton(String name) const</td>
<td></td>
</tr>
<tr>
<td>String localize_path(String path) const</td>
<td></td>
</tr>
<tr>
<td>bool has(String name) const</td>
<td></td>
</tr>
<tr>
<td>bool has_singleton(String name) const</td>
<td></td>
</tr>
<tr>
<td>bool is_persisting(String name) const</td>
<td></td>
</tr>
<tr>
<td>bool load_resource_pack(String pack)</td>
<td></td>
</tr>
<tr>
<td>String localize_path(String path) const</td>
<td></td>
</tr>
<tr>
<td>int save()</td>
<td></td>
</tr>
<tr>
<td>int save_custom(String file)</td>
<td></td>
</tr>
<tr>
<td>void set_order(String name, int pos)</td>
<td></td>
</tr>
<tr>
<td>void set_persisting(String name, bool enable)</td>
<td></td>
</tr>
</tbody>
</table>

12.106.3 Description

Contains global variables accessible from everywhere. Use the normal Object API, such as “Globals.get(variable)”, “Globals.set(variable,value)” or “Globals.has(variable)” to access them. Variables stored in engine.cfg are also loaded into globals, making this object very useful for reading custom game configuration options.

12.106.4 Member Function Description

- void add_property_info(Dictionary hint)

Add a custom property info to a property. The dictionary must contain: name:String, and optionally hint:int), hint_string:String.

Example:

```python
Globals.set("category/property_name", 0)

var property_info = {
    "name": "category/property_name",
    "type": TYPE_INT,
    "hint": PROPERTY_HINT_ENUM,
    "hint_string": "one,two,three"
}

Globals.add_property_info(property_info)
```

- void clear(String name)

Clear the whole configuration (not recommended, may break things).

- int get_order(String name) const

Return the order of a configuration value (influences when saved to the config file).
- **Object** `get_singleton ( String name )` const
- **String** `globalize_path ( String path )` const

Convert a localized path (res://) to a full native OS path.

- **bool** `has ( String name )` const

Return true if a configuration value is present.

- **bool** `has_singleton ( String name )` const
- **bool** `is_persisting ( String name )` const

If returns true, this value can be saved to the configuration file. This is useful for editors.

- **bool** `load_resource_pack ( String pack )`
- **String** `localize_path ( String path )` const

Convert a path to a localized path (res:// path).

- **int** `save ( )`
- **int** `save_custom ( String file )`
- **void** `set_order ( String name, int pos )`

Set the order of a configuration value (influences when saved to the config file).

- **void** `set_persisting ( String name, bool enable )`

If set to true, this value can be saved to the configuration file. This is useful for editors.

### 12.107 GraphEdit

**Inherits:** Control < CanvasItem < Node < Object

**Category:** Core

#### 12.107.1 Brief Description

GraphEdit is an area capable of showing various GraphNodes. It manages connection events between them.

#### 12.107.2 Member Functions

<table>
<thead>
<tr>
<th>Error</th>
<th><code>connect_node ( String from, int from_port, String to, int to_port )</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td><code>disconnect_node ( String from, int from_port, String to, int to_port )</code></td>
</tr>
<tr>
<td>Array</td>
<td><code>get_connection_list ( )</code> const</td>
</tr>
<tr>
<td>Vector2</td>
<td><code>get_scroll_offs ( )</code> const</td>
</tr>
<tr>
<td>float</td>
<td><code>get_zoom ( )</code> const</td>
</tr>
<tr>
<td>bool</td>
<td><code>is_node_connected ( String from, int from_port, String to, int to_port )</code></td>
</tr>
<tr>
<td>bool</td>
<td><code>is_right_disconnects_enabled ( )</code> const</td>
</tr>
<tr>
<td>void</td>
<td><code>set_right_disconnects ( bool enable )</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_zoom ( float p_zoom )</code></td>
</tr>
</tbody>
</table>
12.107.3 Signals

- `_begin_node_move ( )`
  Signal sent at the beginning of a GraphNode movement.
- `_end_node_move ( )`
  Signal sent at the end of a GraphNode movement.
- `connection_request ( String from, int from_slot, String to, int to_slot )`
  Signal sent to the GraphEdit when the connection between ‘from_slot’ slot of ‘from’ GraphNode and ‘to_slot’ slot of ‘to’ GraphNode is attempted to be created.
- `delete_nodes_request ( )`
  Signal sent when a GraphNode is attempted to be removed from the GraphEdit.
- `disconnection_request ( String from, int from_slot, String to, int to_slot )`
  Signal sent to the GraphEdit when the connection between ‘from_slot’ slot of ‘from’ GraphNode and ‘to_slot’ slot of ‘to’ GraphNode is attempted to be removed.
- `duplicate_nodes_request ( )`
  Signal sent when a GraphNode is attempted to be duplicated in the GraphEdit.
- `popup_request ( Vector2 p_position )`
  Signal sent when a popup is requested. Happens on right-clicking in the GraphEdit. ‘p_position’ is the position of the mouse pointer when the signal is sent.

12.107.4 Description

GraphEdit manages the showing of GraphNodes it contains, as well as connections and disconnections between them. Signals are sent for each of these two events. Disconnection between GraphNodes slots is disabled by default.

It is greatly advised to enable low processor usage mode (see `OS.set_low_processor_usage_mode`) when using GraphEdits.

12.107.5 Member Function Description

- `Error connect_node ( String from, int from_port, String to, int to_port )`
  Create a connection between ‘from_port’ slot of ‘from’ GraphNode and ‘to_port’ slot of ‘to’ GraphNode. If the connection already exists, no connection is created.
- `void disconnect_node ( String from, int from_port, String to, int to_port )`
  Remove the connection between ‘from_port’ slot of ‘from’ GraphNode and ‘to_port’ slot of ‘to’ GraphNode, if connection exists.
- `Array get_connection_list ( ) const`
  Return an Array containing the list of connections. A connection consists in a structure of the form `{from_slot: 0, from: “GraphNode name 0”, to_slot: 1, to: “GraphNode name 1” }`
- `Vector2 get_scroll_ofs ( ) const`
  Return the scroll offset.
- `float get_zoom ( ) const`
Return the current zoom value.

- **bool is_node_connected ( String from, int from_port, String to, int to_port )**
  Return true if the ‘from_port’ slot of ‘from’ GraphNode is connected to the ‘to_port’ slot of ‘to’ GraphNode.

- **bool is_right_disconnects_enabled ( ) const**
  Return true is the disconnection of connections is enable in the visual GraphEdit. False otherwise.

- **void set_right_disconnects ( bool enable )**
  Enable the disconnection of existing connections in the visual GraphEdit by left-clicking a connection and releasing into the void.

- **void set_zoom ( float p_zoom )**
  Set the zoom value of the GraphEdit. Zoom value is between 0.01; 1.728.

### 12.108 GraphNode

**Inherits:** Container < Control < CanvasItem < Node < Object

**Category:** Core

#### 12.108.1 Brief Description

A GraphNode is a container with several input and output slots allowing connections between GraphNodes. Slots can have different, incompatible types.
12.108.2 Member Functions

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void clear_all_slots()</td>
<td>Removes all slots from the GraphNode.</td>
</tr>
<tr>
<td>void clear_slot(int idx)</td>
<td>Removes a specific slot by its index.</td>
</tr>
<tr>
<td>Color get_connection_input_color(int idx)</td>
<td>Returns the color of the input slot at index idx.</td>
</tr>
<tr>
<td>int get_connection_input_count()</td>
<td>Returns the count of input slots.</td>
</tr>
<tr>
<td>Vector2 get_connection_input_pos(int idx)</td>
<td>Returns the position of the input slot at index idx.</td>
</tr>
<tr>
<td>int get_connection_input_type(int idx)</td>
<td>Returns the type of the input slot at index idx.</td>
</tr>
<tr>
<td>Color get_connection_output_color(int idx)</td>
<td>Returns the color of the output slot at index idx.</td>
</tr>
<tr>
<td>int get_connection_output_count()</td>
<td>Returns the count of output slots.</td>
</tr>
<tr>
<td>Vector2 get_connection_output_pos(int idx)</td>
<td>Returns the position of the output slot at index idx.</td>
</tr>
<tr>
<td>int get_connection_output_type(int idx)</td>
<td>Returns the type of the output slot at index idx.</td>
</tr>
<tr>
<td>Color get_slot_color_left(int idx)</td>
<td>Returns the color of the left slot at index idx.</td>
</tr>
<tr>
<td>Color get_slot_color_right(int idx)</td>
<td>Returns the color of the right slot at index idx.</td>
</tr>
<tr>
<td>int get_slot_type_left(int idx)</td>
<td>Returns the type of the left slot at index idx.</td>
</tr>
<tr>
<td>int get_slot_type_right(int idx)</td>
<td>Returns the type of the right slot at index idx.</td>
</tr>
<tr>
<td>String get_title()</td>
<td>Returns the title of the GraphNode.</td>
</tr>
<tr>
<td>bool is_close_button_visible()</td>
<td>Returns true if the close button is visible.</td>
</tr>
<tr>
<td>bool is_slot_enabled_left(int idx)</td>
<td>Returns true if the left slot at index idx is enabled.</td>
</tr>
<tr>
<td>bool is_slot_enabled_right(int idx)</td>
<td>Returns true if the right slot at index idx is enabled.</td>
</tr>
<tr>
<td>void set_offset(Vector2 offset)</td>
<td>Sets the offset of the GraphNode.</td>
</tr>
<tr>
<td>void set_show_close_button(bool show)</td>
<td>Sets whether the close button is shown.</td>
</tr>
<tr>
<td>void set_slot(int idx, bool enable_left, int type_left, Color color_left, bool enable_right, int type_right, Color color_right)</td>
<td>Sets the state of a specific slot.</td>
</tr>
<tr>
<td>void set_title(String title)</td>
<td>Sets the title of the GraphNode.</td>
</tr>
</tbody>
</table>

12.108.3 Signals

- **close_request()**
  Signal sent on closing the GraphNode.

- **dragged (Vector2 from, Vector2 to)**
  Signal sent when the GraphNode is dragged.

- **offset_changed()**
  Signal sent when the GraphNode is moved.

- **raise_request()**
  Signal sent when the GraphNode is requested to be displayed over other ones. Happens on focusing (clicking into) the GraphNode.

12.108.4 Description

A GraphNode is a container defined by a title. It can have 1 or more input and output slots, which can be enabled (shown) or disabled (not shown) and have different (incompatible) types. Colors can also be assigned to slots. A tuple of input and output slots is defined for each GUI element included in the GraphNode. Input and output connections are left and right slots, but only enabled slots are counted as connections.
12.108.5 Member Function Description

- **void** clear_all_slots ( )
  Disable all input and output slots of the GraphNode.

- **void** clear_slot ( int idx )
  Disable input and output slot whose index is ‘idx’.

- **Color** get_connection_input_color ( int idx )
  Return the color of the input connection ‘idx’.

- **int** get_connection_input_count ( )
  Return the number of enabled input slots (connections) to the GraphNode.

- **Vector2** get_connection_input_pos ( int idx )
  Return the position of the input connection ‘idx’.

- **int** get_connection_input_type ( int idx )
  Return the type of the input connection ‘idx’.

- **Color** get_connection_output_color ( int idx )
  Return the color of the output connection ‘idx’.

- **int** get_connection_output_count ( )
  Return the number of enabled output slots (connections) of the GraphNode.

- **Vector2** get_connection_output_pos ( int idx )
  Return the position of the output connection ‘idx’.

- **int** get_connection_output_type ( int idx )
  Return the type of the output connection ‘idx’.

- **Vector2** get_offset ( ) const
  Return the offset of the GraphNode.

- **Color** get_slot_color_left ( int idx ) const
  Return the color set to ‘idx’ left (input) slot.

- **Color** get_slot_color_right ( int idx ) const
  Return the color set to ‘idx’ right (output) slot.

- **int** get_slot_type_left ( int idx ) const
  Return the (integer) type of left (input) ‘idx’ slot.

- **int** get_slot_type_right ( int idx ) const
  Return the (integer) type of right (output) ‘idx’ slot.

- **String** get_title ( ) const
  Return the title of the GraphNode.

- **bool** is_close_button_visible ( ) const
  Returns true if the close button is shown. False otherwise.

- **bool** is_slot_enabled_left ( int idx ) const
Return true if left (input) slot ‘idx’ is enabled. False otherwise.

- \texttt{bool is\_slot\_enabled\_right ( int idx )} const

Return true if right (output) slot ‘idx’ is enabled. False otherwise.

- \texttt{void set\_offset ( Vector2 offset )}

Set the offset of the GraphNode.

- \texttt{void set\_show\_close\_button ( bool show )}

Show the close button on the GraphNode if ‘show’ is true (disabled by default). If enabled, a connection on the signal close\_request is needed for the close button to work.

- \texttt{void set\_slot ( int idx, bool enable\_left, int type\_left, Color color\_left, bool enable\_right, int type\_right, Color color\_right )}

Set the tuple of input/output slots defined by ‘idx’ ID. ‘left’ slots are input, ‘right’ are output. ‘type’ is an integer defining the type of the slot. Refer to description for the compatibility between slot types.

- \texttt{void set\_title ( String title )}

Set the title of the GraphNode.

### 12.109 GridContainer

Inherits: \texttt{Container < Control < CanvasItem < Node < Object}}

Category: Core

#### 12.109.1 Brief Description

Grid container used to arrange elements in a grid like layout

#### 12.109.2 Member Functions

<table>
<thead>
<tr>
<th>\texttt{int}</th>
<th>\texttt{get_columns ( ) const}</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{void}</td>
<td>\texttt{set_columns ( int columns )}</td>
</tr>
</tbody>
</table>

#### 12.109.3 Description

Grid container will arrange its children in a grid like structure, the grid columns are specified using the \texttt{set\_columns} method and the number of rows will be equal to the number of children in the container divided by the number of columns, for example: if the container has 5 children, and 2 columns, there will be 3 rows in the container. Notice that grid layout will preserve the columns and rows for every size of the container.

#### 12.109.4 Member Function Description

- \texttt{int get\_columns ( ) const}

Returns the number of columns in this container

- \texttt{void set\_columns ( int columns )}

Sets the numbers of columns in the container, then reorder its children to accommodate the new layout
12.110 GridMap

Inherits: Spatial < Node < Object
Category: Core

12.110.1 Brief Description

12.110.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>AABB</td>
<td>area_get_bounds (int area) const</td>
</tr>
<tr>
<td>String</td>
<td>area_get_name (int area) const</td>
</tr>
<tr>
<td>Color</td>
<td>area_get_portal_disable_color (int area) const</td>
</tr>
<tr>
<td>float</td>
<td>area_get_portal_disable_distance (int area) const</td>
</tr>
<tr>
<td>bool</td>
<td>area_is_exterior_portal (int area) const</td>
</tr>
<tr>
<td>void</td>
<td>area_set_exterior_portal (int area, bool enable)</td>
</tr>
<tr>
<td>void</td>
<td>area_set_name (int area, String name)</td>
</tr>
<tr>
<td>void</td>
<td>area_set_portal_disable_color (int area, Color color)</td>
</tr>
<tr>
<td>void</td>
<td>area_set_portal_disable_distance (int area, float distance)</td>
</tr>
<tr>
<td>void</td>
<td>bake_geometry ()</td>
</tr>
<tr>
<td>void</td>
<td>clear ()</td>
</tr>
<tr>
<td>int</td>
<td>create_area (int id, AABB area)</td>
</tr>
<tr>
<td>void</td>
<td>erase_area (int area)</td>
</tr>
<tr>
<td>int</td>
<td>get_cell_item (int x, int y, int z) const</td>
</tr>
<tr>
<td>int</td>
<td>get_cell_item_orientation (int x, int y, int z) const</td>
</tr>
<tr>
<td>float</td>
<td>get_cell_size () const</td>
</tr>
<tr>
<td>bool</td>
<td>get_center_x () const</td>
</tr>
<tr>
<td>bool</td>
<td>get_center_y () const</td>
</tr>
<tr>
<td>bool</td>
<td>get_center_z () const</td>
</tr>
<tr>
<td>int</td>
<td>get_octant_size () const</td>
</tr>
<tr>
<td>MeshLibrary</td>
<td>get_theme () const</td>
</tr>
<tr>
<td>int</td>
<td>get_unused_area_id () const</td>
</tr>
<tr>
<td>bool</td>
<td>is_baking_enabled () const</td>
</tr>
<tr>
<td>bool</td>
<td>is_using_baked_light () const</td>
</tr>
<tr>
<td>void</td>
<td>resource_changed (Object resource)</td>
</tr>
<tr>
<td>void</td>
<td>set_bake (bool enable)</td>
</tr>
<tr>
<td>void</td>
<td>set_cell_item (int x, int y, int z, int item, int orientation=0)</td>
</tr>
<tr>
<td>void</td>
<td>set_cell_size (float size)</td>
</tr>
<tr>
<td>void</td>
<td>set_center_x (bool enable)</td>
</tr>
<tr>
<td>void</td>
<td>set_center_y (bool enable)</td>
</tr>
<tr>
<td>void</td>
<td>set_center_z (bool enable)</td>
</tr>
<tr>
<td>void</td>
<td>set_clip (bool enabled, bool clipabove=true, int floor=0, int axis=0)</td>
</tr>
<tr>
<td>void</td>
<td>set_octant_size (int size)</td>
</tr>
<tr>
<td>void</td>
<td>set_theme (MeshLibrary theme)</td>
</tr>
<tr>
<td>void</td>
<td>set_use_baked_light (bool use)</td>
</tr>
</tbody>
</table>

12.110.3 Numeric Constants

- INVALID_CELL_ITEM = -1
12.110.4 Member Function Description

- `AABB area_get_bounds (int area) const`
- `String area_get_name (int area) const`
- `Color area_get_portal_disable_color (int area) const`
- `float area_get_portal_disable_distance (int area) const`
- `bool area_is_exterior_portal (int area) const`
- `void area_set_exterior_portal (int area, bool enable)`
- `void area_set_name (int area, String name)`
- `void area_set_portal_disable_color (int area, Color color)`
- `void area_set_portal_disable_distance (int area, float distance)`
- `void bake_geometry ()`
- `void clear ()`
- `int create_area (int id, AABB area)`
- `void erase_area (int area)`
- `int get_cell_item (int x, int y, int z) const`
- `int get_cell_item_orientation (int x, int y, int z) const`
- `float get_cell_size () const`
- `bool get_center_x () const`
- `bool get_center_y () const`
- `bool get_center_z () const`
- `int get_octant_size () const`
- `MeshLibrary get_theme () const`
- `int get_unused_area_id () const`
- `bool is_baking_enabled () const`
- `bool is_using_baked_light () const`
- `void resource_changed (Object resource)`
- `void set_bake (bool enable)`
- `void set_cell_item (int x, int y, int z, int item, int orientation=0)`
- `void set_cell_size (float size)`
- `void set_center_x (bool enable)`
- `void set_center_y (bool enable)`
- `void set_center_z (bool enable)`
- `void set_clip (bool enabled, bool clipabove=true, int floor=0, int axis=0)`
- `void set_octant_size (int size)`
- `void set_theme (MeshLibrary theme)`
- `void set_use_baked_light (bool use)`
12.111 GrooveJoint2D

Inherits: Joint2D < Node2D < CanvasItem < Node < Object

Category: Core

12.111.1 Brief Description

Groove constraint for 2D physics.

12.111.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>float</td>
<td>get_initial_offset() const</td>
</tr>
<tr>
<td>float</td>
<td>get_length() const</td>
</tr>
<tr>
<td>void</td>
<td>set_initial_offset(float offset)</td>
</tr>
<tr>
<td>void</td>
<td>set_length(float length)</td>
</tr>
</tbody>
</table>

12.111.3 Description

Groove constraint for 2D physics. This is useful for making a body “slide” through a segment placed in another.

12.111.4 Member Function Description

- float get_initial_offset() const
  Set the final offset of the groove on body A.

- float get_length() const
  Return the length of the groove.

- void set_initial_offset(float offset)
  Set the initial offset of the groove on body A.

- void set_length(float length)
  Set the length of the groove.

12.112 HBoxContainer

Inherits: BoxContainer < Container < Control < CanvasItem < Node < Object

Category: Core

12.112.1 Brief Description

Horizontal box container.
12.112.2 Description

Horizontal box container. See BoxContainer.

12.113 HButtonArray

**Inherits:** ButtonArray < Control < CanvasItem < Node < Object

Category: Core

12.113.1 Brief Description

Horizontal button array.

12.113.2 Description

Horizontal button array. See ButtonArray.

12.114 HingeJoint

**Inherits:** Joint < Spatial < Node < Object

Category: Core

12.114.1 Brief Description

12.114.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
<th>Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool</td>
<td>get_flag</td>
<td>int flag</td>
</tr>
<tr>
<td>float</td>
<td>get_param</td>
<td>int param</td>
</tr>
<tr>
<td>void</td>
<td>set_flag</td>
<td>int flag, bool enabled</td>
</tr>
<tr>
<td>void</td>
<td>set_param</td>
<td>int param, float value</td>
</tr>
</tbody>
</table>

12.114.3 Numeric Constants

- PARAM_BIAS = 0
- PARAM_LIMIT_UPPER = 1
- PARAM_LIMIT_LOWER = 2
- PARAM_LIMIT_BIAS = 3
- PARAM_LIMIT_SOFTNESS = 4
- PARAM_LIMIT_RELAXATION = 5
- PARAM_MOTOR_TARGET_VELOCITY = 6
- PARAM_MOTOR_MAX_IMPULSE = 7
- PARAM_MAX = 8
• FLAG_USE_LIMIT = 0
• FLAG_ENABLE_MOTOR = 1
• FLAG_MAX = 2

12.114.4 Member Function Description

• bool get_flag ( int flag ) const
• float get_param ( int param ) const
• void set_flag ( int flag, bool enabled )
• void set_param ( int param, float value )

12.115 HScrollBar

Inherits: ScrollBar < Range < Control < CanvasItem < Node < Object
Category: Core

12.115.1 Brief Description

Horizontal scroll bar.

12.115.2 Description

Horizontal scroll bar. See ScrollBar. This one goes from left (min) to right (max).

12.116 HSeparator

Inherits: Separator < Control < CanvasItem < Node < Object
Category: Core

12.116.1 Brief Description

Horizontal separator.

12.116.2 Description

Horizontal separator. See Separator. It is used to separate objects vertically, though (but it looks horizontal!).

12.117 HSlider

Inherits: Slider < Range < Control < CanvasItem < Node < Object
Category: Core
12.117.1 Brief Description

Horizontal slider.

12.117.2 Description

Horizontal slider. See Slider. This one goes from left (min) to right (max).

12.118 HSplitContainer

Inherits: SplitContainer < Container < Control < CanvasItem < Node < Object

Category: Core

12.118.1 Brief Description

Horizontal split container.

12.118.2 Description

Horizontal split container. See SplitContainer. This goes from left to right.

12.119 HTTPClient

Inherits: Reference < Object

Category: Core

12.119.1 Brief Description

Hyper-text transfer protocol client.
### 12.119.2 Member Functions

<table>
<thead>
<tr>
<th>void</th>
<th>close()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error</td>
<td>connect(String host, int port, bool use_ssl=false, bool verify_host=true)</td>
</tr>
<tr>
<td>StreamPeer</td>
<td>get_connection() const</td>
</tr>
<tr>
<td>int</td>
<td>get_response_body_length() const</td>
</tr>
<tr>
<td>int</td>
<td>get_response_code() const</td>
</tr>
<tr>
<td>StringArray</td>
<td>get_response_headers()</td>
</tr>
<tr>
<td>Dictionary</td>
<td>get_response_headers_as_dictionary()</td>
</tr>
<tr>
<td>int</td>
<td>get_status() const</td>
</tr>
<tr>
<td>bool</td>
<td>has_response() const</td>
</tr>
<tr>
<td>bool</td>
<td>is_blocking_mode_enabled() const</td>
</tr>
<tr>
<td>bool</td>
<td>is_response_chunked() const</td>
</tr>
<tr>
<td>Error</td>
<td>poll()</td>
</tr>
<tr>
<td>String</td>
<td>query_string_from_dict(Dictionary fields)</td>
</tr>
<tr>
<td>RawArray</td>
<td>read_response_body_chunk()</td>
</tr>
<tr>
<td>int</td>
<td>request(int method, String url, StringArray headers, String body=&quot;&quot;)</td>
</tr>
<tr>
<td>int</td>
<td>request_raw(int method, String url, StringArray headers, RawArray body)</td>
</tr>
<tr>
<td>int</td>
<td>send_body_data(RawArray body)</td>
</tr>
<tr>
<td>int</td>
<td>send_body_text(String body)</td>
</tr>
<tr>
<td>void</td>
<td>set_blocking_mode(bool enabled)</td>
</tr>
<tr>
<td>void</td>
<td>set_connection(StreamPeer connection)</td>
</tr>
<tr>
<td>void</td>
<td>set_read_chunk_size(int bytes)</td>
</tr>
</tbody>
</table>

### 12.119.3 Numeric Constants

- METHOD_GET = 0
- METHOD_HEAD = 1
- METHOD_POST = 2
- METHOD_PUT = 3
- METHOD_DELETE = 4
- METHOD_OPTIONS = 5
- METHOD_TRACE = 6
- METHOD_CONNECT = 7
- METHOD_MAX = 8
- STATUS_DISCONNECTED = 0
- STATUS_RESOLVING = 1
- STATUS_CANT_RESOLVE = 2
- STATUS_CONNECTING = 3
- STATUS_CANT_CONNECT = 4
- STATUS_CONNECTED = 5
- STATUS_REQUESTING = 6
- STATUS_BODY = 7
- STATUS_CONNECTION_ERROR = 8
• STATUS_SSL_HANDSHAKE_ERROR = 9
• RESPONSE_CONTINUE = 100
• RESPONSE_SWITCHING_PROTOCOLS = 101
• RESPONSE_PROCESSING = 102
• RESPONSE_OK = 200
• RESPONSE_CREATED = 201
• RESPONSE_ACCEPTED = 202
• RESPONSE_NON_AUTHORITATIVE_INFORMATION = 203
• RESPONSE_NO_CONTENT = 204
• RESPONSE_RESET_CONTENT = 205
• RESPONSE_PARTIAL_CONTENT = 206
• RESPONSE_MULTI_STATUS = 207
• RESPONSE_IM_USED = 226
• RESPONSE_MULTIPLE_CHOICES = 300
• RESPONSEMOVED_PERMANENTLY = 301
• RESPONSE_FOUND = 302
• RESPONSESEE_OTHER = 303
• RESPONSE_NOT_MODIFIED = 304
• RESPONSE_USE_PROXY = 305
• RESPONSE_TEMPORARY_REDIRECT = 307
• RESPONSE_BAD_REQUEST = 400
• RESPONSE_UNAUTHORIZED = 401
• RESPONSE_PAYMENT_REQUIRED = 402
• RESPONSE_FORBIDDEN = 403
• RESPONSE_NOT_FOUND = 404
• RESPONSE_METHOD_NOT_ALLOWED = 405
• RESPONSE_NOT_ACCEPTABLE = 406
• RESPONSE_PROXY_AUTHENTICATION_REQUIRED = 407
• RESPONSE_REQUEST_TIMEOUT = 408
• RESPONSE_CONFLICT = 409
• RESPONSE_GONE = 410
• RESPONSE_LENGTH_REQUIRED = 411
• RESPONSE_PRECONDITION_FAILED = 412
• RESPONSE_REQUEST_ENTITY_TOO_LARGE = 413
• RESPONSE_REQUEST_URI_TOO_LONG = 414
• RESPONSE_UNSUPPORTED_MEDIA_TYPE = 415
• RESPONSE_REQUESTED_RANGE_NOT_SATISFIABLE = 416
• RESPONSE_EXPECTATION_FAILED = 417
• RESPONSE_UNPROCESSABLE_ENTITY = 422
• RESPONSE_LOCKED = 423
• RESPONSE_FAILED_DEPENDENCY = 424
• RESPONSE_UPGRADE_REQUIRED = 426
• RESPONSE_INTERNAL_SERVER_ERROR = 500
• RESPONSE_NOT_IMPLEMENTED = 501
• RESPONSE_BAD_GATEWAY = 502
• RESPONSE_SERVICE_UNAVAILABLE = 503
• RESPONSE_GATEWAY_TIMEOUT = 504
• RESPONSE_HTTP_VERSION_NOT_SUPPORTED = 505
• RESPONSE_INSUFFICIENT_STORAGE = 507
• RESPONSE_NOT_EXTENDED = 510

12.119.4 Description

Hyper-text transfer protocol client. Supports SSL and SSL server certificate verification. Can be reused to connect to different hosts and make many requests.

12.119.5 Member Function Description

• void close ( )
  Closes the current connection, allows for reusal of HTTPClient.
  • Error connect ( String host, int port, bool use_ssl=false, bool verify_host=true )
    Connect to a host. This needs to be done before any requests are sent.
    The host should not have http:// prepended but will strip the protocol identifier if provided.
    verify_host will check the SSL identity of the host if set to true.
    • StreamPeer get_connection ( ) const
      Return current connection.
    • int get_response_body_length ( ) const
      Return the response’s body length.
    • int get_response_code ( ) const
      Return the HTTP status code of the response.
    • StringArray get_response_headers ( )
      Return the response headers.
    • Dictionary get_response_headers_as_dictionary ( )
Returns all response headers as dictionary where the case-sensitivity of the keys and values is kept like the server delivers it. A value is a simple String, this string can have more than one value where "; " is used as separator.

Structure: ("key":"value1; value2")

Example: (content-length:12), (Content-Type:application/json; charset=UTF-8)

- **int get_status()** const

  Returns a STATUS_* enum constant. Need to call poll in order to get status updates.

- **bool has_response()** const

  Return whether this **HTTPClient** has a response available.

- **bool is_blocking_mode_enabled()** const

  Return whether blocking mode is enabled.

- **bool is_response_chunked()** const

  Return whether this **HTTPClient** has a response that is chunked.

- **Error poll()**

  This needs to be called in order to have any request processed. Check results with **get_status**

- **String query_string_from_dict(Dictionary fields)**

  Generates a GET/POST application/x-www-form-urlencoded style query string from a provided dictionary, e.g.:

  ```
  var fields = {"username": "user", "password": "pass"}
  String queryString = httpClient.query_string_from_dict(fields)
  returns:= "username=user&password=pass"
  ```

- **RawArray read_response_body_chunk()**

  Reads one chunk from the response.

- **int request(int method, String url, StringArray headers, String body="")**

  Sends a request to the connected host. The url is what is normally behind the hostname, i.e. in http://somehost.com/index.php, url would be “index.php”.

  Headers are HTTP request headers.

  To create a POST request with query strings to push to the server, do:

  ```
  var fields = {"username" : "user", "password" : "pass"}
  var query_string = httpClient.query_string_from_dict(fields)
  var headers = ["Content-Type: application/x-www-form-urlencoded", "Content-Length: ",
                  str(query_string.length())]
  var result = httpClient.request(httpClient.METHOD_POST, "index.php", headers,
                                 query_string)
  ```

- **int request_raw(int method, String url, StringArray headers, RawArray body)**

  Sends a raw request to the connected host. The url is what is normally behind the hostname, i.e. in http://somehost.com/index.php, url would be “index.php”.

  Headers are HTTP request headers.

  Sends body raw, as a byte array, does not encode it in any way.

- **int send_body_data(RawArray body)**

  Stub function
• `int send_body_text ( String body )`

Stub function

• `void set_blocking_mode ( bool enabled )`

If set to true, execution will block until all data is read from the response.

• `void set_connection ( StreamPeer connection )`

Set connection to use, for this client.

• `void set_read_chunk_size ( int bytes )`

Sets the size of the buffer used and maximum bytes to read per iteration. see `read_response_body_chunk`

### 12.120 HTTPRequest

**Inherits:** Node < Object  
**Category:** Core

#### 12.120.1 Brief Description

A Node with the ability to send HTTP requests.

#### 12.120.2 Member Functions

<table>
<thead>
<tr>
<th>void</th>
<th><code>cancel_request</code> ( )</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td><code>get_body_size</code> ( ) const</td>
</tr>
<tr>
<td>int</td>
<td><code>get_body_size_limit</code> ( ) const</td>
</tr>
<tr>
<td>String</td>
<td><code>get_download_file</code> ( ) const</td>
</tr>
<tr>
<td>int</td>
<td><code>get_downloaded_bytes</code> ( ) const</td>
</tr>
<tr>
<td>int</td>
<td><code>get_http_client_status</code> ( ) const</td>
</tr>
<tr>
<td>int</td>
<td><code>get_max_redirects</code> ( ) const</td>
</tr>
<tr>
<td>bool</td>
<td><code>is_using_threads</code> ( ) const</td>
</tr>
<tr>
<td>int</td>
<td><code>request</code> ( String url, StringArray custom_headers=StringArray([]), bool ssl_validate_domain=true, int method=0, String request_data=&quot;&quot; )</td>
</tr>
<tr>
<td>void</td>
<td><code>set_body_size_limit</code> ( int bytes )</td>
</tr>
<tr>
<td>void</td>
<td><code>set_download_file</code> ( String path )</td>
</tr>
<tr>
<td>void</td>
<td><code>set_max_redirects</code> ( int amount )</td>
</tr>
<tr>
<td>void</td>
<td><code>set_use_threads</code> ( bool enable )</td>
</tr>
</tbody>
</table>

#### 12.120.3 Signals

• `request_completed ( int result, int response_code, StringArray headers, RawArray body )`

This signal is emitted upon request completion.
12.120.4 Numeric Constants

- **RESULT_SUCCESS** = 0 — Request successful.
- **RESULT_CHUNKED_BODY_SIZE_MISMATCH** = 1
- **RESULT_CANT_CONNECT** = 2 — Request failed while connecting.
- **RESULT_CANT_RESOLVE** = 3 — Request failed while resolving.
- **RESULT_CONNECTION_ERROR** = 4 — Request failed due to connection(read/write) error.
- **RESULT_SSL_HANDSHAKE_ERROR** = 5 — Request failed on SSL handshake.
- **RESULT_NO_RESPONSE** = 6 — Request does not have a response(yet).
- **RESULT_BODY_SIZE_LIMIT_EXCEEDED** = 7 — Request exceed its maximum size limit, see `set_body_size_limit`.
- **RESULT_REQUEST_FAILED** = 8 — Request failed. (unused)
- **RESULT_DOWNLOAD_FILE_CANT_OPEN** = 9 — HTTPRequest couldn’t open the download file.
- **RESULT_DOWNLOAD_FILE_WRITE_ERROR** = 10 — HTTPRequest couldn’t write to the download file.
- **RESULT_REDIRECT_LIMIT_REACHED** = 11 — Request reached its maximum redirect limit, see `set_max_redirects`.

12.120.5 Description

A Node with the ability to send HTTP requests. Uses a `HTTPClient` internally, supports HTTPS.
Can be used to make HTTP requests or download files via HTTP.

12.120.6 Member Function Description

- **void cancel_request ( )**
  Cancel the current request.
- **int get_body_size ( ) const**
  Return the response body length.
- **int get_body_size_limit ( ) const**
  Return current body size limit.
- **String get_download_file ( ) const**
  Return the file this request will download into.
- **int get_downloaded_bytes ( ) const**
  Return the amount of bytes this HTTPRequest downloaded.
- **int get_http_client_status ( ) const**
  Return the current status of the underlying `HTTPClient`.
- **int get_max_redirects ( ) const**
  Return the maximum amount of redirects that will be followed.
- **bool is_using_threads ( ) const**
Whether this request is using threads.

- `int request ( String url, StringArray custom_headers=StringArray([]), bool ssl_validate_domain=true, int method=0, String request_data="")`

Make a HTTP request (by default GET, unless specified otherwise as method). The url is the complete url including “http://” or “https://” which will be parsed for a host and a port.

The custom_headers are HTTP request headers which will be used. If User-Agent is not specified a Godot specific will be used.

The ssl_validate_domain specifies if in case of HTTPS the server certificate should be verified.

- `void set_body_size_limit ( int bytes )`

Set the response body size limit.

- `void set_download_file ( String path )`

Set the file to download into. Outputs the response body into the file.

- `void set_max_redirects ( int amount )`

Set the maximum amount of redirects the request will follow.

- `void set_use_threads ( bool enable )`

Make this HTTPRequest use threads.

### 12.121 Image

**Category:** Built-In Types

#### 12.121.1 Brief Description

Image datatype.
12.121.2 Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Image ( int width, int height, bool mipmaps, int format )</code></td>
<td>Creates a new image with the given dimensions and format.</td>
</tr>
<tr>
<td><code>blend_rect ( Image src, Rect2 src_rect, Vector2 dest=0 )</code></td>
<td>Blends the source image into the destination image.</td>
</tr>
<tr>
<td><code>blend_rect_mask ( Image src, Image mask, Rect2 src_rect, Vector2 dest=0 )</code></td>
<td>Blends the source image into the destination image using a mask.</td>
</tr>
<tr>
<td><code>blit_rect ( Image src, Rect2 src_rect, Vector2 dest=0 )</code></td>
<td>Copies the source image into the destination image.</td>
</tr>
<tr>
<td><code>blit_rect_mask ( Image src, Image mask, Rect2 src_rect, Vector2 dest=0 )</code></td>
<td>Copies the source image into the destination image using a mask.</td>
</tr>
<tr>
<td><code>brush_transfer ( Image src, Image brush, Vector2 pos=0 )</code></td>
<td>Transfers a brush from one image to another.</td>
</tr>
<tr>
<td><code>brushed ( Image src, Image brush, Vector2 pos=0 )</code></td>
<td>Returns a new image with a brushed effect.</td>
</tr>
<tr>
<td><code>compressed ( int format=0 )</code></td>
<td>Compresses the image.</td>
</tr>
<tr>
<td><code>converted ( int format=0 )</code></td>
<td>Converts the image format.</td>
</tr>
<tr>
<td><code>decompressed ( )</code></td>
<td>Decompresses the image.</td>
</tr>
<tr>
<td><code>empty ( )</code></td>
<td>Returns true if the image is empty.</td>
</tr>
<tr>
<td><code>fill ( Color color=0 )</code></td>
<td>Fills the image with a color.</td>
</tr>
<tr>
<td><code>fix_alpha_edges ( )</code></td>
<td>Fixes alpha edges in the image.</td>
</tr>
<tr>
<td><code>get_data ( )</code></td>
<td>Gets the data from the image.</td>
</tr>
<tr>
<td><code>get_format ( )</code></td>
<td>Returns the format of the image.</td>
</tr>
<tr>
<td><code>get_height ( )</code></td>
<td>Returns the height of the image.</td>
</tr>
<tr>
<td><code>get_pixel ( int x, int y, int mipmap_level=0 )</code></td>
<td>Gets a pixel from the image.</td>
</tr>
<tr>
<td><code>get_rect ( Rect2 area=0 )</code></td>
<td>Gets a rectangle from the image.</td>
</tr>
<tr>
<td><code>get_used_rect ( )</code></td>
<td>Returns the used rectangle of the image.</td>
</tr>
<tr>
<td><code>get_width ( )</code></td>
<td>Returns the width of the image.</td>
</tr>
<tr>
<td><code>load ( String path=0 )</code></td>
<td>Loads an image from a file.</td>
</tr>
<tr>
<td><code>put_pixel ( int x, int y, Color color, int mipmap_level=0 )</code></td>
<td>Puts a pixel into the image.</td>
</tr>
<tr>
<td><code>resized ( int x, int y, int interpolation=1 )</code></td>
<td>Resizes the image.</td>
</tr>
<tr>
<td><code>save_png ( String path=0 )</code></td>
<td>Saves the image as a PNG file.</td>
</tr>
</tbody>
</table>

12.121.3 Numeric Constants

- `COMPRESS_BC = 0`
- `COMPRESS_PVRTC2 = 1`
- `COMPRESS_PVRTC4 = 2`
- `COMPRESS_ETC = 3`
- `FORMAT_GRAYSCALE = 0`
- `FORMAT_INTENSITY = 1`
- `FORMAT_GRAYSCALE_ALPHA = 2`
- `FORMAT_RGB = 3`
- `FORMAT_RGBA = 4`
- `FORMAT_INDEXED = 5`
- `FORMAT_INDEXED_ALPHA = 6`
- `FORMAT_YUV_422 = 7`
- `FORMAT_YUV_444 = 8`
- `FORMAT_BC1 = 9`
- `FORMAT_BC2 = 10`
• FORMAT_BC3 = 11
• FORMAT_BC4 = 12
• FORMAT_BC5 = 13
• FORMAT_PVRTC2 = 14
• FORMAT_PVRTC2_ALPHA = 15
• FORMAT_PVRTC4 = 16
• FORMAT_PVRTC4_ALPHA = 17
• FORMAT_ETC = 18
• FORMAT_ATC = 19
• FORMAT_ATC_ALPHA_EXPLICIT = 20
• FORMAT_ATC_ALPHA_INTERPOLATED = 21
• FORMAT_CUSTOM = 22
• INTERPOLATE_NEAREST = 0
• INTERPOLATE_BILINEAR = 1
• INTERPOLATE_CUBIC = 2

12.121.4 Description

Built in native image datatype. Contains image data, which can be converted to a texture, and several functions to interact with it.

12.121.5 Member Function Description

• Image Image ( int width, int height, bool mipmaps, int format )
Create an empty image of a specific size and format.

• void blend_rect ( Image src, Rect2 src_rect, Vector2 dest=0 )
Alpha-blends a “src_rect” Rect2 from “src” Image to this Image on coordinates “dest”.

• void blend_rect_mask ( Image src, Image mask, Rect2 src_rect, Vector2 dest=0 )
Alpha-blends a “src_rect” Rect2 from “src” Image to this Image using a “mask” Image on coordinates “dest”. Alpha channels are required for both “src” and “mask”, dest pixels and src pixels will blend if the corresponding mask pixel’s alpha value is not 0. “src” Image and “mask” Image *must* have the same size (width and height) but they can have different formats

• void blit_rect ( Image src, Rect2 src_rect, Vector2 dest=0 )
Copy a “src_rect” Rect2 from “src” Image to this Image on coordinates “dest”.

• void blit_rect_mask ( Image src, Image mask, Rect2 src_rect, Vector2 dest=0 )
Blits a “src_rect” Rect2 from “src” Image to this Image using a “mask” Image on coordinates “dest”. Alpha channel is required for “mask”, will copy src pixel onto dest if the corresponding mask pixel’s alpha value is not 0. “src” Image and “mask” Image *must* have the same size (width and height) but they can have different formats

• void brush_transfer ( Image src, Image brush, Vector2 pos=0 )
Transfer data from “src” to this Image using a “brush” as a mask/brush on coordinates “pos”.

12.121. Image
• `Image brushed ( Image src, Image brush, Vector2 pos=0 )`
  Return a new `Image` from this `Image` that is created by brushing see `brush_transfer`.
• `Image compressed ( int format=0 )`
  Return a new compressed `Image` from this `Image` using one of `Image.COMPRESS_*`
• `Image converted ( int format=0 )`
  Return a new `Image` from this `Image` with a different format.
• `Image decompressed ( )`
  Return a new decompressed `Image`.
• `bool empty ( )`
  Return whether this `Image` is empty(no data).
• void `fill ( Color color=0 )`
  Fills an `Image` with a specified `Color`
• void `fix_alpha_edges ( )`
• `RawArray get_data ( )`
  Return the raw data of the `Image`.
• int `get_format ( )`
  Return the format of the `Image`, one of `Image.FORMAT_*`
• int `get_height ( )`
  Return the height of the `Image`.
• `Color get_pixel ( int x, int y, int mipmap_level=0 )`
  Return the color of the pixel in the `Image` on coordinates “x,y” on mipmap level “mipmap_level”.
• `Image get_rect ( Rect2 area=0 )`
  Return a new `Image` that is a copy of “area” in this `Image`.
• `Rect2 get_used_rect ( )`
  Return the area of this `Image` that is used/visibly colored/opaque.
• int `get_width ( )`
  Return the width of the `Image`.
• `int load ( String path=0 )`
  Load an `Image`.
• void `put_pixel ( int x, int y, Color color, int mipmap_level=0 )`
  Put a pixel of “color” on coordinates “x,y” on mipmap level “mipmap_level”.
• `Image resized ( int x, int y, int interpolation=1 )`
  Return a new `Image` from this `Image` that is resized to size “x,y” using `Image.INTERPOLATE_*`.
• `int save_png ( String path=0 )`
  Save this `Image` as a png.
12.122 ImageTexture

Inherits: Texture < Resource < Reference < Object

Category: Core

12.122.1 Brief Description

A Texture based on an Image.

12.122.2 Member Functions

```cpp
void create ( int width, int height, int format, int flags=7 )
void create_from_image ( Image image, int flags=7 )
void fix_alpha_edges ()
Image get_data ( ) const
int get_format ( ) const
float get_lossy_storage_quality ( ) const
int get_storage ( ) const
void load ( String path )
void normal_to_xy ( )
void premultiply_alpha ( )
void set_data ( Image image )
void set_lossy_storage_quality ( float quality )
void set_size_override ( Vector2 size )
void set_storage ( int mode )
void shrink_x2_and_keep_size ( )
```

12.122.3 Numeric Constants

- **STORAGE_RAW = 0** — Image data is stored raw and unaltered.
- **STORAGE_COMPRESS_LOSSY = 1** — Image data is compressed with a lossy algorithm. You can set the storage quality with `set_lossy_storage_quality`.
- **STORAGE_COMPRESS_LOSSLESS = 2** — Image data is compressed with a lossless algorithm.

12.122.4 Description

A Texture based on an Image. Can be created from an Image.

12.122.5 Member Function Description

- void `create ( int width, int height, int format, int flags=7 )`
  
Create a new ImageTexture with “width” and “height”.
  
“format” one of Image.FORMAT_*.
  
“flags” one or more of Texture.FLAG_*.
- void `create_from_image ( Image image, int flags=7 )`
Create a new `ImageTexture` from an `Image` with “flags” from `Texture.FLAG_*`. 

- `void fix_alpha_edges ()` 
- `Image get_data ()` const

Return the `Image` of this `ImageTexture`. 

- `int get_format ()` const

Return the format of the `ImageTexture`, one of `Image.FORMAT_*`. 

- `float get_lossy_storage_quality ()` const

Return the storage quality for `ImageTexture.STORAGE_COMPRESS_LOSSY`. 

- `int get_storage ()` const

Return the storage type. One of `ImageTexture.STORAGE_*`. 

- `void load (String path)` 

Load an ImageTexture. 

- `void normal_to_xy ()` 
- `void premultiply_alpha ()` 
- `void set_data (Image image)` 

Set the `Image` of this `ImageTexture`. 

- `void set_lossy_storage_quality (float quality)` 

Set the storage quality in case of `ImageTexture.STORAGE_COMPRESS_LOSSY`. 

- `void set_size_override (Vector2 size)` 
- `void set_storage (int mode)` 

Set the storage type. One of `ImageTexture.STORAGE_*`. 

- `void shrink_x2_and_keep_size ()`

### 12.123 ImmediateGeometry

**Inherits:** `GeometryInstance < VisualInstance < Spatial < Node < Object`

**Category:** Core

#### 12.123.1 Brief Description

Node to draw simple geometry from code, ala OpenGL 1.x
12.123.2 Member Functions

<table>
<thead>
<tr>
<th>void</th>
<th>add_sphere ( int lats, int lons, float radius, bool add_uv=true )</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>add_vertex ( Vector3 pos )</td>
</tr>
<tr>
<td>void</td>
<td>begin ( int primitive, Texture texture=NULL )</td>
</tr>
<tr>
<td>void</td>
<td>clear ()</td>
</tr>
<tr>
<td>void</td>
<td>end ()</td>
</tr>
<tr>
<td>void</td>
<td>set_color ( Color color )</td>
</tr>
<tr>
<td>void</td>
<td>set_normal ( Vector3 normal )</td>
</tr>
<tr>
<td>void</td>
<td>set_tangent ( Plane tangent )</td>
</tr>
<tr>
<td>void</td>
<td>set_uv ( Vector2 uv )</td>
</tr>
<tr>
<td>void</td>
<td>set_uv2 ( Vector2 uv )</td>
</tr>
</tbody>
</table>

12.123.3 Member Function Description

- void **add_sphere** ( int lats, int lons, float radius, bool add_uv=true )
  Simple helper to draw an uv sphere, with given latitudes, longitude and radius.
- void **add_vertex** ( Vector3 pos )
  Add a vertex with the currently set color/uv/etc.
- void **begin** ( int primitive, Texture texture=NULL )
  Begin drawing (And optionally pass a texture override). When done call end(). For more information on how this works, search for glBegin() glEnd() references.
  For the type of primitive, use the Mesh.PRIMITIVE_* enumerations.
- void **clear** ()
  Clear everything that was drawn using begin/end.
- void **end** ()
  Call this when done adding a batch of geometry, otherwise it can’t be displayed.
- void **set_color** ( Color color )
  Set the color that the next vertex will use to be drawn.
- void **set_normal** ( Vector3 normal )
  Set the normal that the next vertex will use to be drawn.
- void **set_tangent** ( Plane tangent )
  Set the tangent (and binormal facing) that the next vertex will use to be drawn.
- void **set_uv** ( Vector2 uv )
  Set the UV that the next vertex will use to be drawn.
- void **set_uv2** ( Vector2 uv )
  Set the second layer of UV that the next vertex will use to be drawn.
12.124 Input

Inherits: Object

Inherited By: InputDefault

Category: Core

12.124.1 Brief Description

A Singleton that deals with inputs.

12.124.2 Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void action_press (String action)</td>
<td></td>
</tr>
<tr>
<td>void action_release (String action)</td>
<td></td>
</tr>
<tr>
<td>void add_joy_mapping (String mapping, bool update_existing=false)</td>
<td>Adds a mapping for a joystick button.</td>
</tr>
<tr>
<td>Vector3 get_accelerometer ()</td>
<td></td>
</tr>
<tr>
<td>Array get_connected_joysticks ()</td>
<td></td>
</tr>
<tr>
<td>Vector3 get_gravity ()</td>
<td></td>
</tr>
<tr>
<td>Vector3 get_gyroscope ()</td>
<td></td>
</tr>
<tr>
<td>float get_joy_axis (int device, int axis)</td>
<td>Gets the axis value for a joystick.</td>
</tr>
<tr>
<td>int get_joy_axis_index_from_string (String axis)</td>
<td></td>
</tr>
<tr>
<td>String get_joy_axis_string (int axis_index)</td>
<td></td>
</tr>
<tr>
<td>int get_joy_button_index_from_string (String button)</td>
<td></td>
</tr>
<tr>
<td>String get_joy_button_string (int button_index)</td>
<td></td>
</tr>
<tr>
<td>String get_joy_guid (int device)</td>
<td></td>
</tr>
<tr>
<td>String get_joy_name (int device)</td>
<td></td>
</tr>
<tr>
<td>float get_joy_vibration_duration (int device)</td>
<td>Gets the duration of a joystick vibration.</td>
</tr>
<tr>
<td>Vector2 get_joy_vibration_strength (int device)</td>
<td></td>
</tr>
<tr>
<td>Vector3 get_magnetometer ()</td>
<td></td>
</tr>
<tr>
<td>int get_mouse_button_mask () const</td>
<td></td>
</tr>
<tr>
<td>int get_mouse_mode () const</td>
<td></td>
</tr>
<tr>
<td>Vector2 get_mouse_speed () const</td>
<td></td>
</tr>
<tr>
<td>bool is_action_pressed (String action)</td>
<td>Checks if an action is pressed.</td>
</tr>
<tr>
<td>bool is_joy_button_pressed (int device, int button)</td>
<td></td>
</tr>
<tr>
<td>bool is_joy_known (int device)</td>
<td></td>
</tr>
<tr>
<td>bool is_key_pressed (int scancode)</td>
<td></td>
</tr>
<tr>
<td>bool is_mouse_button_pressed (int button)</td>
<td></td>
</tr>
<tr>
<td>void parse_input_event (InputEvent event)</td>
<td>Parses an input event.</td>
</tr>
<tr>
<td>void remove_joy_mapping (String guid)</td>
<td></td>
</tr>
<tr>
<td>void set_custom_mouse_cursor (Texture image, Vector2 hotspot=Vector2(0, 0))</td>
<td>Sets a custom mouse cursor.</td>
</tr>
<tr>
<td>void set_mouse_mode (int mode)</td>
<td></td>
</tr>
<tr>
<td>void start_joy_vibration (int device, float weak_magnitude, float strong_magnitude, float duration=0)</td>
<td>Starts a joystick vibration.</td>
</tr>
<tr>
<td>void stop_joy_vibration (int device)</td>
<td></td>
</tr>
<tr>
<td>void warp_mouse_pos (Vector2 to)</td>
<td></td>
</tr>
</tbody>
</table>

12.124.3 Signals

- joy_connection_changed (int index, bool connected)
Emitted when a joystick device has been connected or disconnected

### 12.124.4 Numeric Constants

- **MOUSE_MODE_VISIBLE** = 0 — Makes the mouse cursor visible if it is hidden.
- **MOUSE_MODE_HIDDEN** = 1 — Makes the mouse cursor hidden if it is visible.
- **MOUSE_MODE_CAPTURED** = 2 — Captures the mouse. The mouse will be hidden and unable to leave the game window. But it will still register movement and mouse button presses.

### 12.124.5 Description

A Singleton that deals with inputs. This includes key presses, mouse buttons and movement, joysticks, and input actions.

### 12.124.6 Member Function Description

- **void action_press** (*String* action)
  This will simulate pressing the specified action.

- **void action_release** (*String* action)
  If the specified action is already pressed, this will release it.

- **void add_joy_mapping** (*String* mapping, *bool* update_existing=false)
  Add a new mapping entry (in SDL2 format) to the mapping database. Optionally update already connected devices.

- **Vector3 get_accelerometer** ()
  If the device has an accelerometer, this will return the movement.

- **Array get_connected_joysticks** ()
  Returns an *Array* containing the device IDs of all currently connected joysticks.

- **Vector3 get_gravity** ()

- **Vector3 get_gyroscope** ()
  If the device has a gyroscope, this will return the rate of rotation in rad/s around a device’s x, y, and z axis.

- **float get_joy_axis** (*int* device, *int* axis)
  Returns the current value of the joystick axis at given index (see JOY_* constants in *Global Scope*)

- **int get_joy_axis_index_from_string** (*String* axis)

- **String get_joy_axis_string** (*int* axis_index)

- **int get_joy_button_index_from_string** (*String* button)

- **String get_joy_button_string** (*int* button_index)

- **String get_joy_guid** (*int* device) const
  Returns a SDL2 compatible device guid on platforms that use gamepad remapping. Returns “Default Gamepad” otherwise.

- **String get_joy_name** (*int* device)
Returns the name of the joystick at the specified device index

- float get_joy_vibration_duration (int device)

Returns the duration of the current vibration effect in seconds.

- Vector2 get_joy_vibration_strength (int device)

Returns the strength of the joystick vibration: x is the strength of the weak motor, and y is the strength of the strong motor.

- Vector3 get_magnetometer()

If the device has a magnetometer, this will return the magnetic field strength in micro-Tesla for all axes.

- int get_mouse_button_mask() const

Returns mouse buttons as a bitmask. If multiple mouse buttons are pressed at the same time the bits are added together.

- int get_magnetometer()

Return the mouse mode. See the constants for more information.

- Vector2 get_mouse_speed() const

Returns the mouse speed for the last time the cursor was moved, and this until the next frame where the mouse moves. This means that even if the mouse is not moving, this function will still return the value of the last motion.

- bool is_action_pressed(String action)

Returns true or false depending on whether the action event is pressed. Actions and their events can be set in the Project Settings / Input Map tab. Or be set with InputMap.

- bool is_joy_button_pressed (int device, int button)

Returns if the joystick button at the given index is currently pressed. (see JOY_* constants in @Global Scope)

- bool is_joy_known (int device)

Returns if the specified device is known by the system. This means that it sets all button and axis indices exactly as defined in the JOY_* constants (see @Global Scope). Unknown joysticks are not expected to match these constants, but you can still retrieve events from them.

- bool is_key_pressed (int scancode)

Returns true or false depending on whether the key is pressed or not. You can pass KEY_*, which are pre-defined constants listed in @Global Scope.

- bool is_mouse_button_pressed (int button)

Returns true or false depending on whether mouse button is pressed or not. You can pass BUTTON_*, which are pre-defined constants listed in @Global Scope.

- void parse_input_event (InputEvent event)

- void remove_joy_mapping (String guid)

Removes all mappings from the internal db that match the given uid.

- void set_custom_mouse_cursor (Texture image, Vector2 hotspot=Vector2(0, 0))

Set a custom mouse cursor image, which is only visible inside the game window. The hotspot can also be specified.

- void set_mouse_mode (int mode)

Set the mouse mode. See the constants for more information.

- void start_joy_vibration (int device, float weak_magnitude, float strong_magnitude, float duration=0)
Starts to vibrate the joystick. Joysticks usually come with two rumble motors, a strong and a weak one. weak_magnitude is the strength of the weak motor (between 0 and 1) and strong_magnitude is the strength of the strong motor (between 0 and 1). duration is the duration of the effect in seconds (a duration of 0 will try to play the vibration indefinitely).

Note that not every hardware is compatible with long effect durations, it is recommended to restart an effect if in need to play it for more than a few seconds.

- void stop_joy_vibration (int device)

  Stops the vibration of the joystick.

- void warp_mouse_pos (Vector2 to)

  Sets the mouse position to the specified vector.

12.125 InputDefault

Inherits: Input < Object

Category: Core

12.125.1 Brief Description

Default implementation of the Input class.

12.125.2 Description

Default implementation of the Input class, used internally by the editor and games for default input management.

12.126 InputEvent

Category: Built-In Types

12.126.1 Brief Description

Built-in input event data.

12.126.2 Member Functions

<table>
<thead>
<tr>
<th>bool</th>
<th>is_action (String action)</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool</td>
<td>is_action_pressed (String action)</td>
</tr>
<tr>
<td>bool</td>
<td>is_action_released (String action)</td>
</tr>
<tr>
<td>bool</td>
<td>is_echo ()</td>
</tr>
<tr>
<td>bool</td>
<td>is_pressed ()</td>
</tr>
<tr>
<td>void</td>
<td>set_as_action (String action, bool pressed)</td>
</tr>
</tbody>
</table>
12.126.3 Member Variables

- `int ID` - Event identifier, positive integer increased at each new event.
- `int device` - Device identifier.
- `int type` - Type of event (one of the [InputEvent] constants).

12.126.4 Numeric Constants

- `NONE = 0` — Empty input event.
- `KEY = 1` — Key event.
- `MOUSE_MOTION = 2` — Mouse motion event.
- `MOUSE_BUTTON = 3` — Mouse button event.
- `JOYSTICK_MOTION = 4` — Joystick motion event.
- `JOYSTICK_BUTTON = 5` — Joystick button event.
- `SCREEN_TOUCH = 6` — Screen touch event.
- `SCREEN_DRAG = 7` — Screen drag event.
- `ACTION = 8` — Pre-defined action event (see InputMap).

12.126.5 Description

Built-in input event data. InputEvent is a built-in engine datatype, given that it’s passed around and used so much. Depending on its type, the members contained can be different, so read the documentation well! Input events can also represent actions (editable from the project settings).

12.126.6 Member Function Description

- `bool is_action ( String action )` 
  Return if this input event matches a pre-defined action, no matter the type.

- `bool is_action_pressed ( String action )` 
  Return whether the given action is being pressed (and is not an echo event for KEY events). Not relevant for the event types MOUSE_MOTION, SCREEN_DRAG and NONE.

- `bool is_action_released ( String action )` 
  Return whether the given action is released (i.e. not pressed). Not relevant for the event types MOUSE_MOTION, SCREEN_DRAG and NONE.

- `bool is_echo ( )` 
  Return if this input event is an echo event (only for events of type KEY, it will return false for other types).

- `bool is_pressed ( )` 
  Return if this input event is pressed. Not relevant for the event types MOUSE_MOTION, SCREEN_DRAG and NONE.

- `void set_as_action ( String action, bool pressed )`
Change the input event to an action event of the given name, regardless of its initial type, with the pressed status passed as argument.

### 12.127 InputEventAction

**Category:** Built-In Types

#### 12.127.1 Brief Description

Built-in input event type for actions.

#### 12.127.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool</td>
<td><code>is_action (String action)</code></td>
</tr>
<tr>
<td>bool</td>
<td><code>is_action_pressed (String action)</code></td>
</tr>
<tr>
<td>bool</td>
<td><code>is_action_released (String action)</code></td>
</tr>
<tr>
<td>bool</td>
<td><code>is_echo ()</code></td>
</tr>
<tr>
<td>bool</td>
<td><code>is_pressed ()</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_as_action (String action, bool pressed)</code></td>
</tr>
</tbody>
</table>

#### 12.127.3 Member Variables

- `int ID` - Event identifier, positive integer increased at each new event.
- `int device` - Device identifier.
- `int type` - Type of event (one of the [InputEvent] constants).

#### 12.127.4 Numeric Constants

- `NONE = 0` — Empty input event.
- `KEY = 1` — Key event.
- `MOUSE_MOTION = 2` — Mouse motion event.
- `MOUSE_BUTTON = 3` — Mouse button event.
- `JOYSTICK_MOTION = 4` — Joystick motion event.
- `JOYSTICK_BUTTON = 5` — Joystick button event.
- `SCREEN_TOUCH = 6` — Screen touch event.
- `SCREEN_DRAG = 7` — Screen drag event.
- `ACTION = 8` — Pre-defined action event (see `InputMap`).

#### 12.127.5 Description

Input event type for actions that extends the global `InputEvent` type.
12.127.6 Member Function Description

- `bool is_action (String action)`
  Return if this input event matches a pre-defined action, i.e. always true for InputEventAction.

- `bool is_action_pressed (String action)`
  Return whether the given action is being pressed.

- `bool is_action_released (String action)`
  Return whether the given action is released (i.e. not pressed).

- `bool is_echo ()`
  Return if this input event is an echo event (only for events of type KEY, i.e. always false for this type).

- `bool is_pressed ()`
  Return if this input event is pressed.

- `void set_as_action (String action, bool pressed)`
  Change the input event to an action event of the given name with the pressed status passed as argument.

12.128 InputEventJoystickButton

Category: Built-In Types

12.128.1 Brief Description

Built-in input event type for joystick button events.

12.128.2 Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>bool is_action (String action)</code></td>
<td>Return if this input event matches a pre-defined action, i.e. always true for InputEventAction.</td>
</tr>
<tr>
<td><code>bool is_action_pressed (String action)</code></td>
<td>Return whether the given action is being pressed.</td>
</tr>
<tr>
<td><code>bool is_action_released (String action)</code></td>
<td>Return whether the given action is released (i.e. not pressed).</td>
</tr>
<tr>
<td><code>bool is_echo ()</code></td>
<td>Return if this input event is an echo event (only for events of type KEY, i.e. always false for this type).</td>
</tr>
<tr>
<td><code>bool is_pressed ()</code></td>
<td>Return if this input event is pressed.</td>
</tr>
<tr>
<td><code>void set_as_action (String action, bool pressed)</code></td>
<td>Change the input event to an action event of the given name with the pressed status passed as argument.</td>
</tr>
</tbody>
</table>

12.128.3 Member Variables

- `int ID` - Event identifier, positive integer increased at each new event.
- `int button_index` - Joystick button identifier, one of the JOY_BUTTON_* constants in [@Global Scope].
- `int device` - Device identifier.
- `bool pressed` - Pressed state of the joystick button.
- `float pressure` - Intensity of the button pressure, ranges from 0 to 1.0.
- `int type` - Type of event (one of the [InputEvent] constants).
12.128.4 Numeric Constants

- **NONE** = 0 — Empty input event.
- **KEY** = 1 — Key event.
- **MOUSE_MOTION** = 2 — Mouse motion event.
- **MOUSE_BUTTON** = 3 — Mouse button event.
- **JOYSTICK_MOTION** = 4 — Joystick motion event.
- **JOYSTICK_BUTTON** = 5 — Joystick button event.
- **SCREEN_TOUCH** = 6 — Screen touch event.
- **SCREEN_DRAG** = 7 — Screen drag event.
- **ACTION** = 8 — Pre-defined action event (see `InputMap`).

12.128.5 Description

Input event type for joystick button events that extends the global `InputEvent` type.

12.128.6 Member Function Description

- `bool is_action (String action)`
  Return if this input event matches a pre-defined action.

- `bool is_action_pressed (String action)`
  Return whether the given action is being pressed.

- `bool is_actionReleased (String action)`
  Return whether the given action is released (i.e. not pressed).

- `bool is_echo ()`
  Return if this input event is an echo event (only for events of type KEY, i.e. always false for this type).

- `bool is_pressed ()`
  Return if this input event is pressed.

- `void set_as_action (String action, bool pressed)`
  Change the input event to an action event of the given name with the pressed status passed as argument.

12.129 InputEventJoystickMotion

Category: Built-In Types

12.129.1 Brief Description

Built-in input event type for joystick motion/axis events.
12.129.2 Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>is_action</code></td>
<td><code>String action</code></td>
</tr>
<tr>
<td><code>is_action_pressed</code></td>
<td><code>String action</code></td>
</tr>
<tr>
<td><code>is_action_released</code></td>
<td><code>String action</code></td>
</tr>
<tr>
<td><code>is_echo</code></td>
<td></td>
</tr>
<tr>
<td><code>is_pressed</code></td>
<td></td>
</tr>
<tr>
<td><code>set_as_action</code></td>
<td><code>String action, bool pressed</code></td>
</tr>
</tbody>
</table>

12.129.3 Member Variables

- `int ID` - Event identifier, positive integer increased at each new event.
- `int axis` - Joystick axis identifier, one of the JOY_AXIS_* constants in [@Global Scope].
- `int device` - Device identifier.
- `int type` - Type of event (one of the [InputEvent] constants).
- `float value` - Position of the axis, ranging from -1.0 to 1.0. A value of 0 means that the axis is in its neutral position.

12.129.4 Numeric Constants

- `NONE = 0` — Empty input event.
- `KEY = 1` — Key event.
- `MOUSE_MOTION = 2` — Mouse motion event.
- `MOUSE_BUTTON = 3` — Mouse button event.
- `JOYSTICK_MOTION = 4` — Joystick motion event.
- `JOYSTICK_BUTTON = 5` — Joystick button event.
- `SCREEN_TOUCH = 6` — Screen touch event.
- `SCREEN_DRAG = 7` — Screen drag event.
- `ACTION = 8` — Pre-defined action event (see InputMap).

12.129.5 Description

Input event type for joystick motion/axis events that extends the global InputEvent type.

12.129.6 Member Function Description

- `bool is_action` ( `String action` )
- `bool is_action_pressed` ( `String action` )

Return whether the given action is being pressed.

- `bool is_action_released` ( `String action` )

Return whether the given action is released (i.e. not pressed).

- `bool is_echo` ( )
Return if this input event is an echo event (only for events of type KEY, i.e. always false for this type).

• `bool is_pressed()`

Return if this input event is pressed.

• `void set_as_action(String action, bool pressed)`

Change the input event to an action event of the given name with the pressed status passed as argument.

## 12.130 InputEventKey

**Category:** Built-In Types

### 12.130.1 Brief Description

Built-in input event type for keyboard events.

### 12.130.2 Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>bool is_action(String action)</code></td>
<td></td>
</tr>
<tr>
<td><code>bool is_action_pressed(String action)</code></td>
<td></td>
</tr>
<tr>
<td><code>bool is_action_released(String action)</code></td>
<td></td>
</tr>
<tr>
<td><code>bool is_echo()</code></td>
<td></td>
</tr>
<tr>
<td><code>bool is_pressed()</code></td>
<td></td>
</tr>
<tr>
<td><code>void set_as_action(String action, bool pressed)</code></td>
<td></td>
</tr>
</tbody>
</table>

### 12.130.3 Member Variables

- `int ID` - Event identifier, positive integer increased at each new event.
- `bool alt` - State of the Alt modifier.
- `bool control` - State of the Ctrl modifier.
- `int device` - Device identifier.
- `bool echo` - Echo state of the key, i.e. whether it's a repeat event or not.
- `bool meta` - State of the Meta modifier.
- `bool pressed` - Pressed state of the key.
- `int scancode` - Scancode of the key, one of the KEY_* constants in [@Global Scope].
- `bool shift` - State of the Shift modifier.
- `int type` - Type of event (one of the [InputEvent] constants).
- `int unicode` - Unicode identifier of the key (when relevant).
12.130.4 Numeric Constants

- **NONE** = 0 — Empty input event.
- **KEY** = 1 — Key event.
- **MOUSE_MOTION** = 2 — Mouse motion event.
- **MOUSE_BUTTON** = 3 — Mouse button event.
- **JOYSTICK_MOTION** = 4 — Joystick motion event.
- **JOYSTICK_BUTTON** = 5 — Joystick button event.
- **SCREEN_TOUCH** = 6 — Screen touch event.
- **SCREEN_DRAG** = 7 — Screen drag event.
- **ACTION** = 8 — Pre-defined action event (see *InputMap*).

12.130.5 Description

Input event type for keyboard events that extends the global *InputEvent* type.

12.130.6 Member Function Description

- **bool is_action (String action)**
  Return if this input event matches a pre-defined action.
- **bool is_action_pressed (String action)**
  Return whether the given action is being pressed.
- **bool is_action_released (String action)**
  Return whether the given action is released (i.e. not pressed).
- **bool is_echo ()**
  Return if this input event is an echo event.
- **bool is_pressed ()**
  Return if this input event is pressed.
- **void set_as_action (String action, bool pressed)**
  Change the input event to an action event of the given name with the pressed status passed as argument.

12.131 InputEventMouseButton

Category: Built-In Types

12.131.1 Brief Description

Built-in input event type for mouse button events.
12.131.2 Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool is_action(String action)</td>
<td>Checks if a specific action is active.</td>
</tr>
<tr>
<td>bool is_action_pressed(String action)</td>
<td>Checks if a specific action is pressed.</td>
</tr>
<tr>
<td>bool is_action_released(String action)</td>
<td>Checks if a specific action is released.</td>
</tr>
<tr>
<td>bool is_echo()</td>
<td>Checks if echo is active.</td>
</tr>
<tr>
<td>bool is_pressed()</td>
<td>Checks if a mouse button is pressed.</td>
</tr>
<tr>
<td>void set_as_action(String action, bool pressed)</td>
<td>Sets an action to be pressed.</td>
</tr>
</tbody>
</table>

12.131.3 Member Variables

- **int ID** - Event identifier, positive integer increased at each new event.
- **bool alt** - State of the Alt modifier.
- **int button_index** - Mouse button identifier, one of the BUTTON_* or BUTTON_WHEEL_* constants in [@Global Scope].
- **int button_mask** - Mouse button mask identifier, one of or a bitwise combination of the BUTTON_MASK_* constants in [@Global Scope].
- **bool control** - State of the Control modifier.
- **int device** - Device identifier.
- **bool doubleclick** - Whether the event is a double-click.
- **Vector2 global_pos** - Global position of the mouse click.
- **int global_x** - Global X coordinate of the mouse click.
- **int global_y** - Global Y coordinate of the mouse click.
- **bool meta** - State of the Meta modifier.
- **Vector2 pos** - Local position of the mouse click.
- **bool pressed** - Pressed state of the mouse button.
- **bool shift** - State of the Shift modifier.
- **int type** - Type of event (one of the [InputEvent] constants).
- **int x** - Local X coordinate of the mouse click.
- **int y** - Local Y coordinate of the mouse click.

12.131.4 Numeric Constants

- **NONE** = 0 — Empty input event.
- **KEY** = 1 — Key event.
- **MOUSE_MOTION** = 2 — Mouse motion event.
- **MOUSE_BUTTON** = 3 — Mouse button event.
- **JOYSTICK_MOTION** = 4 — Joystick motion event.
- **JOYSTICK_BUTTON** = 5 — Joystick button event.
- **SCREEN_TOUCH** = 6 — Screen touch event.
• **SCREEN_DRAG** = 7 — Screen drag event.
• **ACTION** = 8 — Pre-defined action event (see *InputMap*).

### 12.131.5 Description

Input event type for mouse button events that extends the global *InputEvent* type.

### 12.131.6 Member Function Description

- **bool is_action (String action)**
  Return if this input event matches a pre-defined action.

- **bool is_action_pressed (String action)**
  Return whether the given action is being pressed.

- **bool is_action_released (String action)**
  Return whether the given action is released (i.e. not pressed).

- **bool is_echo ()**
  Return if this input event is an echo event (only for events of type KEY, i.e. always false for this type).

- **bool is_pressed ()**
  Return if this input event is pressed.

- **void set_as_action (String action, bool pressed)**
  Change the input event to an action event of the given name with the pressed status passed as argument.

### 12.132 InputEventMouseMotion

**Category:** Built-In Types

### 12.132.1 Brief Description

Built-in input event type for mouse motion events.

### 12.132.2 Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>bool is_action (String action)</strong></td>
<td>Return if this input event matches a pre-defined action.</td>
</tr>
<tr>
<td><strong>bool is_action_pressed (String action)</strong></td>
<td>Return whether the given action is being pressed.</td>
</tr>
<tr>
<td><strong>bool is_action_released (String action)</strong></td>
<td>Return whether the given action is released (i.e. not pressed).</td>
</tr>
<tr>
<td><strong>bool is_echo ()</strong></td>
<td>Return if this input event is an echo event (only for events of type KEY, i.e. always false for this type).</td>
</tr>
<tr>
<td><strong>bool is_pressed ()</strong></td>
<td>Return if this input event is pressed.</td>
</tr>
<tr>
<td><strong>void set_as_action (String action, bool pressed)</strong></td>
<td>Change the input event to an action event of the given name with the pressed status passed as argument.</td>
</tr>
</tbody>
</table>
12.132.3 Member Variables

- `int ID` - Event identifier, positive integer increased at each new event.
- `bool alt` - State of the Alt modifier.
- `int button_mask` - Mouse button mask identifier, one of or a bitwise combination of the BUTTON_MASK_* constants in [@Global Scope].
- `bool control` - State of the Ctrl modifier.
- `int device` - Device identifier.
- `Vector2 global_pos` - Global position of the mouse pointer.
- `int global_x` - Global X coordinate of the mouse pointer.
- `int global_y` - Global Y coordinate of the mouse pointer.
- `bool meta` - State of the Meta modifier.
- `Vector2 pos` - Local position of the mouse pointer.
- `Vector2 relative_pos` - Position of the mouse pointer relative to the previous mouse position.
- `int relative_x` - X coordinate of the mouse pointer relative to the previous mouse position.
- `int relative_y` - Y coordinate of the mouse pointer relative to the previous mouse position.
- `bool shift` - State of the Shift modifier.
- `Vector2 speed` - Speed of the mouse pointer.
- `float speed_x` - Speed of the mouse pointer on the X axis.
- `float speed_y` - Speed of the mouse pointer on the Y axis.
- `int type` - Type of event (one of the [InputEvent] constants).
- `int x` - Local X coordinate of the mouse pointer.
- `int y` - Local Y coordinate of the mouse pointer.

12.132.4 Numeric Constants

- `NONE = 0` — Empty input event.
- `KEY = 1` — Key event.
- `MOUSE_MOTION = 2` — Mouse motion event.
- `MOUSE_BUTTON = 3` — Mouse button event.
- `JOYSTICK_MOTION = 4` — Joystick motion event.
- `JOYSTICK_BUTTON = 5` — Joystick button event.
- `SCREEN_TOUCH = 6` — Screen touch event.
- `SCREEN_DRAG = 7` — Screen drag event.
- `ACTION = 8` — Pre-defined action event (see `InputMap`).

12.132.5 Description

Input event type for mouse motion events that extends the global `InputEvent` type.
12.132.6 Member Function Description

- **bool is_action (String action)**
  Return if this input event matches a pre-defined action.

- **bool is_action_pressed (String action)**
  Return whether the given action is being pressed. Not relevant for MOUSE_MOTION events, always false.

- **bool is_action_released (String action)**
  Return whether the given action is released (i.e. not pressed). Not relevant for MOUSE_MOTION events, can be true or false depending on whether is_action is true.

- **bool is_echo ()**
  Return if this input event is an echo event (only for events of type KEY, i.e. always false for this type).

- **bool is_pressed ()**
  Return if this input event is pressed. Not relevant for MOUSE_MOTION events, always false.

- **void set_as_action (String action, bool pressed)**
  Change the input event to an action event of the given name with the (irrelevant for this type) pressed status passed as argument.

12.133 InputEventScreenDrag

Category: Built-In Types

12.133.1 Brief Description

Built-in input event type for screen drag events.

12.133.2 Member Functions

<table>
<thead>
<tr>
<th>bool</th>
<th>is_action (String action)</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool</td>
<td>is_action_pressed (String action)</td>
</tr>
<tr>
<td>bool</td>
<td>is_action_released (String action)</td>
</tr>
<tr>
<td>bool</td>
<td>is_echo ()</td>
</tr>
<tr>
<td>bool</td>
<td>is_pressed ()</td>
</tr>
<tr>
<td>void</td>
<td>set_as_action (String action, bool pressed)</td>
</tr>
</tbody>
</table>

12.133.3 Member Variables

- **int ID** - Event identifier, positive integer increased at each new event.
- **int device** - Device identifier.
- **int index** - Drag event index in the case of a multi-drag event.
- **Vector2 pos** - Position of the drag event.
- **Vector2 relative_pos** - Position of the drag event relative to its start position.
• float relative_x - X coordinate of the drag event relative to its start position.
• float relative_y - Y coordinate of the drag event relative to its start position.
• Vector2 speed - Speed of the drag event.
• float speed_x - Speed of the drag event on the X axis.
• float speed_y - Speed of the drag event on the Y axis.
• int type - Type of event (one of the [InputEvent] constants).
• float x - X coordinate of the drag event.
• float y - Y coordinate of the drag event.

12.133.4 Numeric Constants

• NONE = 0 — Empty input event.
• KEY = 1 — Key event.
• MOUSE_MOTION = 2 — Mouse motion event.
• MOUSE_BUTTON = 3 — Mouse button event.
• JOYSTICK_MOTION = 4 — Joystick motion event.
• JOYSTICK_BUTTON = 5 — Joystick button event.
• SCREEN_TOUCH = 6 — Screen touch event.
• SCREEN_DRAG = 7 — Screen drag event.
• ACTION = 8 — Pre-defined action event (see InputMap).

12.133.5 Description

Input event type for screen drag events that extends the global InputEvent type.

12.133.6 Member Function Description

• bool is_action ( String action )
  Return if this input event matches a pre-defined action.
• bool is_action_pressed ( String action )
  Return whether the given action is being pressed. Not relevant for SCREEN_DRAG events, always false.
• bool is_action_released ( String action )
  Return whether the given action is released (i.e. not pressed). Not relevant for SCREEN_DRAG events, can be true or false depending on whether is_action is true.
• bool is_echo ( )
  Return if this input event is an echo event (only for events of type KEY, i.e. always false for this type).
• bool is_pressed ( )
  Return if this input event is pressed. Not relevant for SCREEN_DRAG events, always false.
• void set_as_action ( String action, bool pressed )
  // Set action
Godot Engine Documentation, Release 2.1

Change the input event to an action event of the given name with the (irrelevant for this type) pressed status passed as argument.

12.134 InputEventScreenTouch

Category: Built-In Types

12.134.1 Brief Description

Built-in input event type for touchscreen drag events.

12.134.2 Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool is_action(String action)</td>
<td></td>
</tr>
<tr>
<td>bool is_action_pressed(String action)</td>
<td></td>
</tr>
<tr>
<td>bool is_action_released(String action)</td>
<td></td>
</tr>
<tr>
<td>bool is_echo()</td>
<td></td>
</tr>
<tr>
<td>bool is_pressed()</td>
<td></td>
</tr>
<tr>
<td>void set_as_action(String action, bool pressed)</td>
<td></td>
</tr>
</tbody>
</table>

12.134.3 Member Variables

- int ID - Event identifier, positive integer increased at each new event.
- int device - Device identifier.
- int index - Touch event index in the case of a multi-touch event.
- Vector2 pos - Position of the touch event.
- bool pressed - Pressed state of the touch event.
- int type - Type of event (one of the [InputEvent] constants).
- float x - X coordinate of the touch event.
- float y - Y coordinate of the touch event.

12.134.4 Numeric Constants

- NONE = 0 — Empty input event.
- KEY = 1 — Key event.
- MOUSE_MOTION = 2 — Mouse motion event.
- MOUSE_BUTTON = 3 — Mouse button event.
- JOYSTICK_MOTION = 4 — Joystick motion event.
- JOYSTICK_BUTTON = 5 — Joystick button event.
- SCREEN_TOUCH = 6 — Screen touch event.
- SCREEN_DRAG = 7 — Screen drag event.
• **ACTION** = 8 — Pre-defined action event (see *InputMap*).

### 12.134.5 Description

Input event type for touchscreen drag events that extends the global *InputEvent* type.

### 12.134.6 Member Function Description

- **bool is_action** *(String action)*
  Return if this input event matches a pre-defined action.
- **bool is_action_pressed** *(String action)*
  Return whether the given action is being pressed.
- **bool is_action_released** *(String action)*
  Return whether the given action is released (i.e. not pressed).
- **bool is_echo** *( )
  Return if this input event is an echo event (only for events of type KEY, i.e. always false for this type).
- **bool is_pressed** *( )
  Return if this input event is pressed.
- **void set_as_action** *(String action, bool pressed)*
  Change the input event to an action event of the given name with the pressed status passed as argument.

### 12.135 InputMap

**Inherits:** *Object*

**Category:** Core

#### 12.135.1 Brief Description

Singleton that manages actions.
12.135.2 Member Functions

<table>
<thead>
<tr>
<th>function</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void action_add_event(String action, InputEvent event)</code></td>
<td>Add an <code>InputEvent</code> to an action. This <code>InputEvent</code> will trigger the action.</td>
</tr>
<tr>
<td><code>void action_erase_event(String action, InputEvent event)</code></td>
<td>Remove an <code>InputEvent</code> from an action.</td>
</tr>
<tr>
<td><code>bool action_has_event(String action, InputEvent event)</code></td>
<td>Whether an action has an <code>InputEvent</code> associated with it.</td>
</tr>
<tr>
<td><code>void add_action(String action)</code></td>
<td>Add an (empty) action to the <code>InputMap</code>. An <code>InputEvent</code> can then be added to this action with <code>action_add_event</code>.</td>
</tr>
<tr>
<td><code>void erase_action(String action)</code></td>
<td>Remove an action from the <code>InputMap</code>.</td>
</tr>
<tr>
<td><code>bool event_is_action(InputEvent event, String action)</code></td>
<td>Return whether the given event is part of an existing action.</td>
</tr>
<tr>
<td><code>String get_action_from_id(int id)</code></td>
<td>Return the action corresponding to the identifier.</td>
</tr>
<tr>
<td><code>int get_action_id(String action)</code></td>
<td>Return the identifier of the given action.</td>
</tr>
<tr>
<td><code>Array get_action_list(String action)</code></td>
<td>Return an array of <code>InputEvents</code> associated with a given action.</td>
</tr>
<tr>
<td><code>Array get_actions()</code></td>
<td>Return an array of all actions in the <code>InputMap</code>.</td>
</tr>
</tbody>
</table>

12.135.3 Description

Singleton that manages actions. `InputMap` has a list of the actions used in `InputEvent`, which can be modified.

12.135.4 Member Function Description

- `void action_add_event(String action, InputEvent event)`
  Add an `InputEvent` to an action. This `InputEvent` will trigger the action.
- `void action_erase_event(String action, InputEvent event)`
  Remove an `InputEvent` from an action.
- `bool action_has_event(String action, InputEvent event)`
  Whether an action has an `InputEvent` associated with it.
- `void add_action(String action)`
  Add an (empty) action to the `InputMap`. An `InputEvent` can then be added to this action with `action_add_event`.
- `void erase_action(String action)`
  Remove an action from the `InputMap`.
- `bool event_is_action(InputEvent event, String action)`
  Return whether the given event is part of an existing action.
- `String get_action_from_id(int id)`
  Return the action corresponding to the identifier.
- `int get_action_id(String action)`
  Return the identifier of the given action.
- `Array get_action_list(String action)`
  Return an array of `InputEvents` associated with a given action.
- `Array get_actions()`
  Return an array of all actions in the `InputMap`.
- `bool has_action(String action)`
  Return whether the given action is an existing action.
Whether this InputMap has a registered action with the given name.

- void load_from_globals ( )

Clear the InputMap and load it anew from Globals.

## 12.136 InstancePlaceholder

**Inherits:** Node < Object  
**Category:** Core

### 12.136.1 Brief Description

Placeholder for the root Node of a PackedScene.

### 12.136.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function Name</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>get_instance_path ( )</td>
<td></td>
</tr>
<tr>
<td>Dictionary</td>
<td>get_stored_values ( bool with_order=false )</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>replace_by_instance ( PackedScene custom_scene=NULL )</td>
<td></td>
</tr>
</tbody>
</table>

### 12.136.3 Description

Turning on the option Load As Placeholder for an instanced scene in the editor causes it to be replaced by an InstancePlaceholder when running the game. This makes it possible to delay actually loading the scene until calling replace_by_instance. This is useful to avoid loading large scenes all at once by loading parts of it selectively.

The InstancePlaceholder does not have a transform. This causes any child nodes to be positioned relatively to the Viewport from point (0,0), rather than their parent as displayed in the editor. Replacing the placeholder with a scene with a transform will transform children relatively to their parent again.

### 12.136.4 Member Function Description

- **String get_instance_path ( ) const**
  Retrieve the path to the PackedScene resource file that is loaded by default when calling replace_by_instance.

- **Dictionary get_stored_values ( bool with_order=false )**

- **void replace_by_instance ( PackedScene custom_scene=NULL )**
  Replace this placeholder by the scene handed as an argument, or the original scene if no argument is given. As for all resources, the scene is loaded only if it’s not loaded already. By manually loading the scene beforehand, delays caused by this function can be avoided.

## 12.137 int

**Category:** Built-In Types
12.137.1 Brief Description

Integer built-in type.

12.137.2 Member Functions

<table>
<thead>
<tr>
<th>int</th>
<th>int (bool from)</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>int (float from)</td>
</tr>
<tr>
<td>int</td>
<td>int (String from)</td>
</tr>
</tbody>
</table>

12.137.3 Description

Integer built-in type.

12.137.4 Member Function Description

- int int (bool from)
  Cast a bool value to an integer value, int(true) will be equals to 1 and int(false) will be equals to 0.

- int int (float from)
  Cast a float value to an integer value, this method simply removes the number fractions, so for example int(2.7) will be equals to 2, int(.1) will be equals to 0 and int(-2.7) will be equals to -2.

- int int (String from)
  Cast a String value to an integer value, this method is an integer parser from a string, so calling this method with an invalid integer string will return 0, a valid string will be something like '1.7'. This method will ignore all non-number characters, so calling int('1e3') will return 13.

12.138 IntArray

Category: Built-In Types

12.138.1 Brief Description

Integer Array.
12.138.2 Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>IntArray Array from</code></td>
<td>Create from a generic array.</td>
</tr>
<tr>
<td><code>void append (int integer)</code></td>
<td>Append an element at the end of the array (alias of <code>push_back</code>).</td>
</tr>
<tr>
<td><code>void append_array (IntArray array)</code></td>
<td>Append an <code>IntArray</code> at the end of this array.</td>
</tr>
<tr>
<td><code>int insert (int idx, int integer)</code></td>
<td>Insert a new int at a given position in the array. The position must be valid, or at the end of the array (pos==size()).</td>
</tr>
<tr>
<td><code>void invert ()</code></td>
<td>Reverse the order of the elements in the array (so first element will now be the last).</td>
</tr>
<tr>
<td><code>void push_back (int integer)</code></td>
<td>Append a value to the array.</td>
</tr>
<tr>
<td><code>void remove (int idx)</code></td>
<td>Remove an element from the array by index.</td>
</tr>
<tr>
<td><code>void resize (int idx)</code></td>
<td>Set the size of the <code>IntArray</code>. If larger than the current size it will reserve some space beforehand, and if it is smaller it will cut off the array.</td>
</tr>
<tr>
<td><code>void set (int idx, int integer)</code></td>
<td>Change the int at the given index.</td>
</tr>
<tr>
<td><code>int size ()</code></td>
<td>Return the array size.</td>
</tr>
</tbody>
</table>

12.138.3 Description

Integer Array. Array of integers. Can only contain integers. Optimized for memory usage, can’t fragment the memory.

12.138.4 Member Function Description

- `IntArray IntArray (Array from)`
  Create from a generic array.
  - `void append (int integer)`
    Append an element at the end of the array (alias of `push_back`).
  - `void append_array (IntArray array)`
    Append an `IntArray` at the end of this array.
  - `int insert (int idx, int integer)`
    Insert a new int at a given position in the array. The position must be valid, or at the end of the array (pos==size()).
  - `void invert ()`
    Reverse the order of the elements in the array (so first element will now be the last).
  - `void push_back (int integer)`
    Append a value to the array.
  - `void remove (int idx)`
    Remove an element from the array by index.
  - `void resize (int idx)`
    Set the size of the `IntArray`. If larger than the current size it will reserve some space beforehand, and if it is smaller it will cut off the array.
  - `void set (int idx, int integer)`
    Change the int at the given index.
  - `int size ()`
    Return the array size.
12.139 InterpolatedCamera

Inherits: Camera < Spatial < Node < Object
Category: Core

12.139.1 Brief Description

12.139.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Member Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>float</td>
<td>get_speed() const</td>
</tr>
<tr>
<td>NodePath</td>
<td>get_target_path() const</td>
</tr>
<tr>
<td>bool</td>
<td>is_interpolation_enabled() const</td>
</tr>
<tr>
<td>void</td>
<td>set_interpolation_enabled(bool target_path)</td>
</tr>
<tr>
<td>void</td>
<td>set_speed(float speed)</td>
</tr>
<tr>
<td>void</td>
<td>set_target(Camera target)</td>
</tr>
<tr>
<td>void</td>
<td>set_target_path(NodePath target_path)</td>
</tr>
</tbody>
</table>

12.139.3 Member Function Description

- float get_speed() const
- NodePath get_target_path() const
- bool is_interpolation_enabled() const
- void set_interpolation_enabled(bool target_path)
- void set_speed(float speed)
- void set_target(Camera target)
- void set_target_path(NodePath target_path)

12.140 InverseKinematics

Inherits: Spatial < Node < Object
Category: Core
12.140.1 Brief Description

12.140.2 Member Functions

<table>
<thead>
<tr>
<th>String</th>
<th>get_bone_name ( ) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>get_chain_size ( ) const</td>
</tr>
<tr>
<td>int</td>
<td>get_iterations ( ) const</td>
</tr>
<tr>
<td>float</td>
<td>get_precision ( ) const</td>
</tr>
<tr>
<td>float</td>
<td>get_speed ( ) const</td>
</tr>
<tr>
<td>void</td>
<td>set_bone_name (String ik_bone)</td>
</tr>
<tr>
<td>void</td>
<td>set_chain_size (int chain_size)</td>
</tr>
<tr>
<td>void</td>
<td>set_iterations (int iterations)</td>
</tr>
<tr>
<td>void</td>
<td>set_precision (float precision)</td>
</tr>
<tr>
<td>void</td>
<td>set_speed (float speed)</td>
</tr>
</tbody>
</table>

12.140.3 Member Function Description

- String get_bone_name ( ) const
- int get_chain_size ( ) const
- int get_iterations ( ) const
- float get_precision ( ) const
- float get_speed ( ) const
- void set_bone_name (String ik_bone)
- void set_chain_size (int chain_size)
- void set_iterations (int iterations)
- void set_precision (float precision)
- void set_speed (float speed)

12.141 IP

Inherits: Object
Inherited By: IP_Unix
Category: Core

12.141.1 Brief Description

IP Protocol support functions.
12.141.2 Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void clear_cache ( String arg0=&quot;&quot; )</td>
<td>Erase a queue ID, removing it from the queue if needed.</td>
</tr>
<tr>
<td>void erase_resolve_item ( int id )</td>
<td>Return a resolved item address, or an empty string if an error happened or</td>
</tr>
<tr>
<td></td>
<td>resolution didn’t happen yet (see get_resolve_item_status).</td>
</tr>
<tr>
<td>Array get_local_addresses ( ) const</td>
<td>Return the status of hostname queued for resolving, given its queue ID.</td>
</tr>
<tr>
<td>String get_resolve_item_address ( int id )</td>
<td>Return status can be any of the RESOLVER_STATUS_* enumeration.</td>
</tr>
<tr>
<td>int get_resolve_item_status ( int id ) const</td>
<td></td>
</tr>
<tr>
<td>String resolve_hostname ( String host, int ip_type=3 )</td>
<td></td>
</tr>
<tr>
<td>int resolve_hostname_queue_item ( String host, int ip_type=3 )</td>
<td></td>
</tr>
</tbody>
</table>

12.141.3 Numeric Constants

- RESOLVER_STATUS_NONE = 0
- RESOLVER_STATUS_WAITING = 1
- RESOLVER_STATUS_DONE = 2
- RESOLVER_STATUS_ERROR = 3
- RESOLVER_MAX_QUERIES = 32
- RESOLVER_INVALID_ID = -1
- TYPE_NONE = 0
- TYPE_IPV4 = 1
- TYPE_IPV6 = 2
- TYPE_ANY = 3

12.141.4 Description

IP contains some support functions for the IPv4 protocol. TCP/IP support is in different classes (see StreamPeerTCP and TCP_Server). IP provides hostname resolution support, both blocking and threaded.

12.141.5 Member Function Description

- void clear_cache ( String arg0="" )
- void erase_resolve_item ( int id )
- Array get_local_addresses ( ) const
- String get_resolve_item_address ( int id ) const
- int get_resolve_item_status ( int id ) const
- String resolve_hostname ( String host, int ip_type=3 )
- int resolve_hostname_queue_item ( String host, int ip_type=3 )
Resolve a given hostname, blocking. Resolved hostname is returned as an IPv4 or IPv6 depending on “ip_type”.

- int resolve_hostname_queue_item ( String host, int ip_type=3 )

Create a queue item for resolving a given hostname to an IPv4 or IPv6 depending on “ip_type”. The queue ID is returned, or RESOLVER_INVALID_ID on error.

### 12.142 IP_Unix

Inherits: IP < Object  
Category: Core

### 12.142.1 Brief Description

### 12.143 ItemList

Inherits: Control < CanvasItem < Node < Object  
Category: Core

### 12.143.1 Brief Description

Control that provides a list of selectable items (and/or icons) in a single column, or optionally in multiple columns.

### 12.143.2 Member Functions

| void | add_icon_item ( Texture icon, bool selectable=true ) |
| void | add_item ( String text, Texture icon=NULL, bool selectable=true ) |
| void | clear ( ) |
| bool | ensure_current_is_visible ( ) |
| int | get_allow_rmb_select ( ) const |
| int | get_fixed_column_width ( ) const |
| Vector2 | get_fixed_icon_size ( ) const |
| int | get_icon_mode ( ) const |
| float | get_icon_scale ( ) const |
| int | get_item_at_pos ( Vector2 pos, bool exact=false ) const |
| int | get_item_count ( ) const |
| Color | get_item_custom_bg_color ( int idx ) const |
| Texture | get_item_icon ( int idx ) const |
| Rect2 | get_item_icon_region ( int idx ) const |
| void | get_item_metadata ( int idx ) const |
| String | get_item_text ( int idx ) const |
| String | get_item_tooltip ( int idx ) const |
| int | get_max_columns ( ) const |
| int | get_max_text_lines ( ) const |
| int | get_select_mode ( ) const |
| IntArray | get_selected_items ( ) |

Continued on next page
12.143.3 Signals

- **item_activated ( int index )**
  Fired when specified list item is activated via double click or Enter.

- **item_rmb_selected ( int index, Vector2 atpos )**
  Fired when specified list item has been selected via right mouse clicking.
  The click position is also provided to allow appropriate popup of context menus at the correct location.

- **item_selected ( int index )**
  Fired when specified item has been selected.

- **multi_selected ( int index, bool selected )**
  Fired when a multiple selection is altered on a list allowing multiple selection.

12.143.4 Numeric Constants

- **ICON_MODE_TOP = 0**
• ICON_MODE_LEFT = 1
• SELECT_SINGLE = 0
• SELECT_MULTI = 1

12.143.5 Description

This control provides a selectable list of items that may be in a single (or multiple columns) with option of text, icons, or both text and icon. Tooltips are supported and may be different for every item in the list. Selectable items in the list may be selected or deselected and multiple selection may be enabled. Selection with right mouse button may also be enabled to allow use of popup context menus. Items may also be ‘activated’ with a double click (or Enter key).

12.143.6 Member Function Description

• void add_icon_item ( Texture icon, bool selectable=true )
  Adds an item to the item list with no text, only an icon.

• void add_item ( String text, Texture icon=NULL, bool selectable=true )
  Adds an item to the item list with specified text. Specify an icon of null for a list item with no icon.
  If selectable is true the list item will be selectable.

• void clear ( )
  Remove all items from the list.

• void ensure_current_is_visible ( )
  Ensure selection is visible, adjusting the scroll position as necessary.

• bool get_allow_rmb_select ( ) const
  Return whether or not items may be selected via right mouse clicking.

• int get_fixed_column_width ( ) const
  If column size has been fixed to a value, return that value.

• Vector2 get_fixed_icon_size ( ) const

• int get_icon_mode ( ) const

• float get_icon_scale ( ) const

• int get_item_at_pos ( Vector2 pos, bool exact=false ) const
  Given a position within the control return the item (if any) at that point.

• int get_item_count ( ) const
  Return count of items currently in the item list.

• Color get_item_custom_bg_color ( int idx ) const

• Texture get_item_icon ( int idx ) const

• Rect2 get_item_icon_region ( int idx ) const

• void get_item_metadata ( int idx ) const
• **String** get_item_text (int idx) const

Return the text for specified item index.

• **String** get_item_tooltip (int idx) const

Return tooltip hint for specified item index.

• int get_max_columns () const

Return total number of columns in use by the list.

• int get_max_text_lines () const

Return total number of lines currently in use by the list.

• int get_select_mode () const

• IntArray get_selected_items ()

Returns the list of selected indexes.

• **Object** get_v_scroll ()

Returns the current vertical scroll bar for the List.

• bool is_item_disabled (int idx) const

Returns whether or not the item at the specified index is disabled

• bool is_itemSelectable (int idx) const

Returns whether or not the item at the specified index is selectable.

• bool is_itemTooltip_enabled (int idx) const

Returns whether the tooltip is enabled for specified item index.

• int is_same_column_width () const

Returns whether or not all columns of the list are of the same size.

• bool is_selected (int idx) const

Returns whether or not item at the specified index is currently selected.

• void remove_item (int idx)

Remove item at specified index from the list.

• void select (int idx, bool single=true)

Select the item at the specified index.

Note: This method does not trigger the item selection signal.

• void set_allow_rmb_select (bool allow)

Allow (or disallow) selection of (selectable) items in the list using right mouse button.

• void set_fixed_column_width (int width)

Set the size (width) all columns in the list are to use.

• void set_fixed_icon_size (Vector2 size)

• void set_icon_mode (int mode)

• void set_icon_scale (float scale)

• void set_item_custom_bg_color (int idx, Color custom_bg_color)
• void set_item_disabled ( int idx, bool disabled )
Disable (or enable) item at specified index.
Disabled items are not be selectable and do not fire activation (Enter or double-click) signals.

• void set_item_icon ( int idx, Texture icon )
Set (or replace) icon of the item at the specified index.

• void set_item_icon_region ( int idx, Rect2 rect )
• void set_item_metadata ( int idx, Variant metadata )
Sets a value (of any type) to be stored with the item at the specified index.

• void set_item_selectable ( int idx, bool selectable )
Allow or disallow selection of the item at the specified index.

• void set_item_text ( int idx, String text )
Sets text of item at specified index.

• void set_item_tooltip ( int idx, String tooltip )
Sets tooltip hint for item at specified index.

• void set_item_tooltip_enabled ( int idx, bool enable )
Sets whether the tooltip is enabled for specified item index.

• void set_max_columns ( int amount )
Set maximum number of columns to use for the list.

• void set_max_text_lines ( int lines )
Set maximum number of lines to use for the list.

• void set_same_column_width ( bool enable )
Sets a fixed size (width) to use for all columns of the list.

• void set_select_mode ( int mode )
• void sort_items_by_text ( )
Sorts items in the list by their text.

• void unselect ( int idx )
Ensure item at specified index is not selected.

12.144 Joint

Inherits: Spatial < Node < Object

Inherited By: ConeTwistJoint, SliderJoint, Generic6DOFJoint, HingeJoint, PinJoint

Category: Core
12.144.1 Brief Description

12.144.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool</td>
<td>get_exclude_nodes_from_collision ( ) const</td>
</tr>
<tr>
<td>NodePath</td>
<td>get_node_a ( ) const</td>
</tr>
<tr>
<td>NodePath</td>
<td>get_node_b ( ) const</td>
</tr>
<tr>
<td>int</td>
<td>get_solver_priority ( ) const</td>
</tr>
<tr>
<td>void</td>
<td>set_exclude_nodes_from_collision ( bool enable )</td>
</tr>
<tr>
<td>void</td>
<td>set_node_a ( NodePath node )</td>
</tr>
<tr>
<td>void</td>
<td>set_node_b ( NodePath node )</td>
</tr>
<tr>
<td>void</td>
<td>set_solver_priority ( int priority )</td>
</tr>
</tbody>
</table>

12.144.3 Member Function Description

- bool get_exclude_nodes_from_collision ( ) const
- NodePath get_node_a ( ) const
- NodePath get_node_b ( ) const
- int get_solver_priority ( ) const
- void set_exclude_nodes_from_collision ( bool enable )
- void set_node_a ( NodePath node )
- void set_node_b ( NodePath node )
- void set_solver_priority ( int priority )

12.145 Joint2D

Inherits: Node2D < CanvasItem < Node < Object

Inherited By: PinJoint2D, DampedSpringJoint2D, GrooveJoint2D

Category: Core

12.145.1 Brief Description

Base node for all joint constraints in 2D physics.

12.145.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>float</td>
<td>get_bias ( ) const</td>
</tr>
<tr>
<td>bool</td>
<td>get_exclude_nodes_from_collision ( ) const</td>
</tr>
<tr>
<td>NodePath</td>
<td>get_node_a ( ) const</td>
</tr>
<tr>
<td>NodePath</td>
<td>get_node_b ( ) const</td>
</tr>
<tr>
<td>void</td>
<td>set_bias ( float bias )</td>
</tr>
<tr>
<td>void</td>
<td>set_exclude_nodes_from_collision ( bool enable )</td>
</tr>
<tr>
<td>void</td>
<td>set_node_a ( NodePath node )</td>
</tr>
<tr>
<td>void</td>
<td>set_node_b ( NodePath node )</td>
</tr>
</tbody>
</table>
12.145.3 Description

Base node for all joint constraints in 2D physics. Joints take 2 bodies and apply a custom constraint.

12.145.4 Member Function Description

- float get_bias ( ) const
- bool get_exclude_nodes_from_collision ( ) const
- NodePath get_node_a ( ) const
  Return the path to the A node for the joint.
- NodePath get_node_b ( ) const
  Return the path to the B node for the joint.
- void set_bias ( float bias )
- void set_exclude_nodes_from_collision ( bool enable )
- void set_node_a ( NodePath node )
  Set the path to the A node for the joint. Must be of type PhysicsBody2D.
- void set_node_b ( NodePath node )
  Set the path to the B node for the joint. Must be of type PhysicsBody2D.

12.146 KinematicBody

Inherits: PhysicsBody < CollisionObject < Spatial < Node < Object

Category: Core

12.146.1 Brief Description

Kinematic body 3D node.
### 12.146.2 Member Functions

<table>
<thead>
<tr>
<th>bool</th>
<th>can_collide_with_character_bodies() const</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool</td>
<td>can_collide_with_kinematic_bodies() const</td>
</tr>
<tr>
<td>bool</td>
<td>can_collide_with_rigid_bodies() const</td>
</tr>
<tr>
<td>bool</td>
<td>can_collide_with_static_bodies() const</td>
</tr>
<tr>
<td>bool</td>
<td>can_teleport_to(Vector3 position)</td>
</tr>
<tr>
<td>Variant</td>
<td>get_collider() const</td>
</tr>
<tr>
<td>int</td>
<td>get_collider_shape() const</td>
</tr>
<tr>
<td>Vector3</td>
<td>get_collider_velocity() const</td>
</tr>
<tr>
<td>float</td>
<td>get_collision_margin() const</td>
</tr>
<tr>
<td>Vector3</td>
<td>get_collision_normal() const</td>
</tr>
<tr>
<td>Vector3</td>
<td>get_collision_pos() const</td>
</tr>
<tr>
<td>bool</td>
<td>is_colliding() const</td>
</tr>
<tr>
<td>Vector3</td>
<td>move(Vector3 rel_vec)</td>
</tr>
<tr>
<td>Vector3</td>
<td>move_to(Vector3 position)</td>
</tr>
<tr>
<td>void</td>
<td>set_collide_with_character_bodies(bool enable)</td>
</tr>
<tr>
<td>void</td>
<td>set_collide_with_kinematic_bodies(bool enable)</td>
</tr>
<tr>
<td>void</td>
<td>set_collide_with_rigid_bodies(bool enable)</td>
</tr>
<tr>
<td>void</td>
<td>set_collide_with_static_bodies(bool enable)</td>
</tr>
<tr>
<td>void</td>
<td>set_collision_margin(float pixels)</td>
</tr>
</tbody>
</table>

### 12.146.3 Description

Kinematic bodies are special types of bodies that are meant to be user-controlled. They are not affected by physics at all (to other types of bodies, such a character or a rigid body, these are the same as a static body). They have however, two main uses:

Simulated Motion: When these bodies are moved manually, either from code or from an AnimationPlayer (with process mode set to fixed), the physics will automatically compute an estimate of their linear and angular velocity. This makes them very useful for moving platforms or other AnimationPlayer-controlled objects (like a door, a bridge that opens, etc).

Kinematic Characters: KinematicBody also has an api for moving objects (the move method) while performing collision tests. This makes them really useful to implement characters that collide against a world, but that don’t require advanced physics.

### 12.146.4 Member Function Description

- **bool can_collide_with_character_bodies() const**
  Return if this body can collide with character bodies.

- **bool can_collide_with_kinematic_bodies() const**
  Return if this body can collide with kinematic bodies.

- **bool can_collide_with_rigid_bodies() const**
  Return if this body can collide with rigid bodies.

- **bool can_collide_with_static_bodies() const**
  Return if this body can collide with static bodies.

- **bool can_teleport_to(Vector3 position)**
Returns whether the KinematicBody can be teleported to the destination given as an argument, checking all collision shapes of the body against potential colliders at the destination.

- **Variant** `get_collider()` const
  Return the body that collided with this one.

- **int** `get_collider_shape()` const
  Return the shape index from the body that collided with this one. If there is no collision, this method will return 0, so collisions must be checked first with `is_colliding`.

- **Vector3** `get_collider_velocity()` const
  Return the velocity of the body that collided with this one.

- **float** `get_collision_margin()` const
  Return the collision margin for this object.

- **Vector3** `get_collision_normal()` const
  Return the normal of the surface the body collided with. This is useful to implement sliding along a surface.

- **Vector3** `get_collision_pos()` const
  Return the point in space where the body is touching another. If there is no collision, this method will return (0,0,0), so collisions must be checked first with `is_colliding`.

- **bool** `is_colliding()` const
  Return whether the body is colliding with another.

- **Vector3** `move(Vector3 rel_vec)`
  Move the body in the given direction, stopping if there is an obstacle. The returned vector is how much movement was remaining before being stopped.

- **Vector3** `move_to(Vector3 position)`
  Move the body to the given position. This is not a teleport, and the body will stop if there is an obstacle. The returned vector is how much movement was remaining before being stopped.

- **void** `set_collide_with_character_bodies(bool enable)`
  Set if this body should collide with character bodies.

- **void** `set_collide_with_kinematic_bodies(bool enable)`
  Set if this body should collide with kinematic bodies.

- **void** `set_collide_with_rigid_bodies(bool enable)`
  Set if this body should collide with rigid bodies.

- **void** `set_collide_with_static_bodies(bool enable)`
  Set if this body should collide with static bodies.

- **void** `set_collision_margin(float pixels)`
  Set the collision margin for this object. A collision margin is an amount that all shapes will grow when computing collisions, to account for numerical imprecision.
12.147 KinematicBody2D

Inherits: PhysicsBody2D < CollisionObject2D < Node2D < CanvasItem < Node < Object

Category: Core

12.147.1 Brief Description

Kinematic body 2D node.

12.147.2 Member Functions

<table>
<thead>
<tr>
<th>Variant</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variant</td>
<td>get_collider ( ) const</td>
</tr>
<tr>
<td>Variant</td>
<td>get_collider_metadata ( ) const</td>
</tr>
<tr>
<td>int</td>
<td>get_collider_shape ( ) const</td>
</tr>
<tr>
<td>Vector2</td>
<td>get_collider_velocity ( ) const</td>
</tr>
<tr>
<td>float</td>
<td>get_collision_margin ( ) const</td>
</tr>
<tr>
<td>Vector2</td>
<td>get_collision_normal ( ) const</td>
</tr>
<tr>
<td>Vector2</td>
<td>get_collision_pos ( ) const</td>
</tr>
<tr>
<td>Array</td>
<td>get_move_and_slide_colliders ( ) const</td>
</tr>
<tr>
<td>Vector2</td>
<td>get_travel ( ) const</td>
</tr>
<tr>
<td>bool</td>
<td>is_colliding ( ) const</td>
</tr>
<tr>
<td>bool</td>
<td>is_move_and_slide_on_ceiling ( ) const</td>
</tr>
<tr>
<td>bool</td>
<td>is_move_and_slide_on_floor ( ) const</td>
</tr>
<tr>
<td>bool</td>
<td>is_move_and_slide_on_wall ( ) const</td>
</tr>
<tr>
<td>Vector2</td>
<td>move ( Vector2 rel_vec )</td>
</tr>
<tr>
<td>Vector2</td>
<td>move_and_slide ( Vector2 linear_velocity, Vector2 floor_normal=Vector2(0, 0), float slope_stop_min_velocity=5, int max_bounces=4 )</td>
</tr>
<tr>
<td>Vector2</td>
<td>move_to ( Vector2 position )</td>
</tr>
<tr>
<td>void</td>
<td>revert_motion ( )</td>
</tr>
<tr>
<td>void</td>
<td>set_collision_margin ( float pixels )</td>
</tr>
<tr>
<td>bool</td>
<td>test_move ( Vector2 rel_vec )</td>
</tr>
<tr>
<td>bool</td>
<td>test_move_from ( Matrix32 from, Vector2 rel_vec )</td>
</tr>
</tbody>
</table>

12.147.3 Description

Kinematic bodies are special types of bodies that are meant to be user-controlled. They are not affected by physics at all (to other types of bodies, such a character or a rigid body, these are the same as a static body). They have however, two main uses:

Simulated Motion: When these bodies are moved manually, either from code or from an AnimationPlayer (with process mode set to fixed), the physics will automatically compute an estimate of their linear and angular velocity.
This makes them very useful for moving platforms or other AnimationPlayer-controlled objects (like a door, a bridge that opens, etc).

Kinematic Characters: KinematicBody2D also has an api for moving objects (the *move* method) while performing collision tests. This makes them really useful to implement characters that collide against a world, but that don’t require advanced physics.

**12.147.4 Member Function Description**

- **Variant** *get_collider* ( ) const
  Return the body that collided with this one.

- **Variant** *get_collider_metadata* ( ) const
  Return the metadata of the shape that collided with this body. If there is no collision, it will return 0, so collisions must be checked first with *is_colliding*. Additionally, this metadata can not be set with *Object.set_meta*, it must be set with *Physics2DServer.body_set_shape_metadata*.

- **int** *get_collider_shape* ( ) const
  Return the shape index from the body that collided with this one. If there is no collision, this method will return 0, so collisions must be checked first with *is_colliding*.

- **Vector2** *get_collider_velocity* ( ) const
  Return the velocity of the body that collided with this one.

- **float** *get_collision_margin* ( ) const
  Return the collision margin for this object.

- **Vector2** *get_collision_normal* ( ) const
  Return the normal of the surface the body collided with. This is useful to implement sliding along a surface.

- **Vector2** *get_collision_pos* ( ) const
  Return the point in space where the body is touching another. If there is no collision, this method will return (0,0), so collisions must be checked first with *is_colliding*.

- **Array** *get_move_and_slide_colliders* ( ) const
  Return the last movement done by the body.

- **bool** *is_colliding* ( ) const
  Return whether the body is colliding with another.

- **bool** *is_move_and_slide_on_ceiling* ( ) const

- **bool** *is_move_and_slide_on_floor* ( ) const

- **bool** *is_move_and_slide_on_wall* ( ) const

- **Vector2** *move* ( Vector2 *rel_vec* ) const
  Move the body in the given direction, stopping if there is an obstacle. The returned vector is how much movement was remaining before being stopped.

- **Vector2** *move_to* ( Vector2 *position* ) const
Move the body to the given position. This is not a teleport, and the body will stop if there is an obstacle. The returned vector is how much movement was remaining before being stopped.

- **void revert_motion()**
  Undo the last movement done by the body.

- **void set_collision_margin(float pixels)**
  Set the collision margin for this object. A collision margin is an amount (in pixels) that all shapes will grow when computing collisions, to account for numerical imprecision.

- **bool test_move(Vector2 rel_vec)**
  Return true if there would be a collision if the body moved in the given direction.

- **bool test_move_from(Matrix32 from, Vector2 rel_vec)**

## 12.148 Label

**Inherits:** Control < CanvasItem < Node < Object

**Category:** Core

### 12.148.1 Brief Description

Control that displays formatted text.

### 12.148.2 Member Functions

```
int get_align() const
int get_line_count() const
int get_line_height() const
int get_lines_skipped() const
int get_max_lines_visible() const
float get_percent_visible() const
String get_text() const
int get_total_character_count() const
int get_valign() const
int get_visible_characters() const
int get_visible_line_count() const
bool has_autowrap() const
bool is_clipping_text() const
bool is_uppercase() const
void set_align(int align)
void set_autowrap(bool enable)
void set_clip_text(bool enable)
void set_lines_skipped(int lines_skipped)
void set_max_lines_visible(int lines_visible)
void set_percent_visible(float percent_visible)
void set_text(String text)
void set_uppercase(bool enable)
void set_valign(int valign)
void set_visible_characters(int amount)
```
12.148.3 Numeric Constants

- ALIGN_LEFT = 0 — Align rows to the left (default).
- ALIGN_CENTER = 1 — Align rows centered.
- ALIGN_RIGHT = 2 — Align rows to the right (default).
- ALIGN_FILL = 3 — Expand row whitespaces to fit the width.
- VALIGN_TOP = 0 — Align the whole text to the top.
- VALIGN_CENTER = 1 — Align the whole text to the center.
- VALIGN_BOTTOM = 2 — Align the whole text to the bottom.
- VALIGN_FILL = 3 — Align the whole text by spreading the rows.

12.148.4 Description

Label is a control that displays formatted text, optionally autowrapping it to the Control area. It inherits from range to be able to scroll wrapped text vertically.

12.148.5 Member Function Description

- int get_align () const
  Return the alignment mode (any of the ALIGN_* enumeration values).
- int get_line_count () const
  Return the amount of lines.
- int get_line_height () const
  Return the height of a line.
- int get_lines_skipped () const
  Return the number of lines to skipped before displaying.
- int get_max_lines_visible () const
  Return the restricted number of lines to display. Returns -1 if unrestricted.
- float get_percent_visible () const
  Return the restricted number of characters to display (as a percentage of the total text).
- String get_text () const
  Return the label text. Text can contain newlines.
- int get_total_character_count () const
  Return the total length of the text.
- int get_valign () const
  Return the vertical alignment mode (any of the VALIGN_* enumeration values).
- int get_visible_characters () const
  Return the restricted number of characters to display. Returns -1 if unrestricted.
- int get_visible_line_count () const
• bool has_autowrap ( ) const
Return the state of the autowrap mode (see set_autowrap).
• bool is_clipping_text ( ) const
Return true if text would be cut off if it is too wide.
• bool is_uppercase ( ) const
Return true if text is displayed in all capitals.
• void set_align ( int align )
Sets the alignment mode to any of the ALIGN_* enumeration values.
• void set_autowrap ( bool enable )
Set autowrap mode. When enabled, autowrap will fit text to the control width, breaking sentences when they exceed the available horizontal space. When disabled, the label minimum width becomes the width of the longest row, and the minimum height large enough to fit all rows.
• void set_clip_text ( bool enable )
Cuts off the rest of the text if it is too wide.
• void set_lines_skipped ( int lines_skipped )
Sets the number of lines to skip before displaying. Useful for scrolling text.
• void set_max_lines_visible ( int lines_visible )
Restricts the number of lines to display. Set to -1 to disable.
• void set_percent_visible ( float percent_visible )
Restricts the number of characters to display (as a percentage of the total text).
• void set_text ( String text )
Set the label text. Text can contain newlines.
• void set_uppercase ( bool enable )
Display text in all capitals.
• void set_valign ( int valign )
Sets the vertical alignment mode to any of the VALIGN_* enumeration values.
• void set_visible_characters ( int amount )
Restricts the number of characters to display. Set to -1 to disable.

12.149 LargeTexture

Inherits: Texture < Resource < Reference < Object
Category: Core

12.149.1 Brief Description

A Texture capable of storing many smaller Textures with offsets.
12.149.2 Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>int add_piece</td>
<td>int add_piece (Vector2 ofs, Texture texture)</td>
</tr>
<tr>
<td>void clear</td>
<td>void clear ()</td>
</tr>
<tr>
<td>int get_piece_count</td>
<td>int get_piece_count () const</td>
</tr>
<tr>
<td>Vector2 get_piece_offset</td>
<td>Vector2 get_piece_offset (int idx) const</td>
</tr>
<tr>
<td>Texture get_piece_texture</td>
<td>Texture get_piece_texture (int idx) const</td>
</tr>
<tr>
<td>void set_piece_offset</td>
<td>void set_piece_offset (int idx, Vector2 ofs)</td>
</tr>
<tr>
<td>void set_piece_texture</td>
<td>void set_piece_texture (int idx, Texture texture)</td>
</tr>
<tr>
<td>void set_size</td>
<td>void set_size (Vector2 size)</td>
</tr>
</tbody>
</table>

12.149.3 Description

A Texture capable of storing many smaller Textures with offsets. You can dynamically add pieces(Textures) to this LargeTexture using different offsets.

12.149.4 Member Function Description

- **int add_piece (Vector2 ofs, Texture texture)**
  Add another Texture to this LargeTexture, starting on offset “ofs”.

- **void clear ()**
  Clear the LargeTexture.

- **int get_piece_count () const**
  Return the number of pieces currently in this LargeTexture.

- **Vector2 get_piece_offset (int idx) const**
  Return the offset of the piece with index “idx”.

- **Texture get_piece_texture (int idx) const**
  Return the Texture of the piece with index “idx”.

- **void set_piece_offset (int idx, Vector2 ofs)**
  Set the offset of the piece with index “idx” to “ofs”.

- **void set_piece_texture (int idx, Texture texture)**
  Set the Texture of the piece with index “idx” to “ofs”.

- **void set_size (Vector2 size)**
  Set the size of this LargeTexture.

12.150 Light

Inherits: VisualInstance < Spatial < Node < Object

Inherited By: SpotLight, OmniLight, DirectionalLight

Category: Core
12.150.1 Brief Description

Provides a base class for different kinds of light nodes.

12.150.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>get_bake_mode() const</td>
</tr>
<tr>
<td>Color</td>
<td>get_color(int color) const</td>
</tr>
<tr>
<td>int</td>
<td>get_operator() const</td>
</tr>
<tr>
<td>float</td>
<td>get_parameter(int variable) const</td>
</tr>
<tr>
<td>Texture</td>
<td>get_projector() const</td>
</tr>
<tr>
<td>bool</td>
<td>has_project_shadows() const</td>
</tr>
<tr>
<td>bool</td>
<td>is_editor_only() const</td>
</tr>
<tr>
<td>bool</td>
<td>is_enabled() const</td>
</tr>
<tr>
<td>void</td>
<td>set_bake_mode(int bake_mode)</td>
</tr>
<tr>
<td>void</td>
<td>set_color(int color, Color value)</td>
</tr>
<tr>
<td>void</td>
<td>set_editor_only(bool editor_only)</td>
</tr>
<tr>
<td>void</td>
<td>set_enabled(bool enabled)</td>
</tr>
<tr>
<td>void</td>
<td>set_operator(int operator)</td>
</tr>
<tr>
<td>void</td>
<td>set_parameter(int variable, float value)</td>
</tr>
<tr>
<td>void</td>
<td>set_project_shadows(bool enable)</td>
</tr>
<tr>
<td>void</td>
<td>set_projector(Texture projector)</td>
</tr>
</tbody>
</table>

12.150.3 Numeric Constants

- PARAM_RADIUS = 2
- PARAM_ENERGY = 3
- PARAM_ATTENUATION = 4
- PARAM_SPOT_ANGLE = 1
- PARAM_SPOT_ATTENUATION = 0
- PARAM_SHADOW_DARKENING = 5
- PARAM_SHADOW_Z_OFFSET = 6
- COLOR_DIFFUSE = 0
- COLOR_SPECULAR = 1
- BAKE_MODE_DISABLED = 0
- BAKE_MODE INDIRECT = 1
- BAKE_MODE INDIRECT_AND_SHADOWS = 2
- BAKE_MODE FULL = 3

12.150.4 Description

Light is the abstract base class for light nodes, so it shouldn’t be used directly (It can’t be instanced). Other types of light nodes inherit from it. Light contains the common variables and parameters used for lighting.
12.150.5 Member Function Description

- `int get_bake_mode () const`
- `Color get_color ( int color ) const`
- `int get_operator () const`
- `float get_parameter ( int variable ) const`
- `Texture get_projector () const`
- `bool has_project_shadows () const`
- `bool is_editor_only () const`
- `bool is_enabled () const`
- `void set_bake_mode ( int bake_mode )`
- `void set_color ( int color, Color value )`
- `void set_editor_only ( bool editor_only )`
- `void set_enabled ( bool enabled )`
- `void set_operator ( int operator )`
- `void set_parameter ( int variable, float value )`
- `void set_project_shadows ( bool enable )`
- `void set_projector ( Texture projector )`

12.151 Light2D

Inherits: `Node2D < CanvasItem < Node < Object`

Category: Core

12.151.1 Brief Description

Node that casts light in a 2D environment.

12.151.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color</td>
<td><code>get_color () const</code></td>
</tr>
<tr>
<td>float</td>
<td><code>get_energy () const</code></td>
</tr>
<tr>
<td>float</td>
<td><code>get_height () const</code></td>
</tr>
<tr>
<td>int</td>
<td><code>get_item_mask () const</code></td>
</tr>
<tr>
<td>int</td>
<td><code>get_item_shadow_mask () const</code></td>
</tr>
<tr>
<td>int</td>
<td><code>get_layer_range_max () const</code></td>
</tr>
<tr>
<td>int</td>
<td><code>get_layer_range_min () const</code></td>
</tr>
<tr>
<td>int</td>
<td><code>get_mode () const</code></td>
</tr>
<tr>
<td>int</td>
<td><code>get_shadow_buffer_size () const</code></td>
</tr>
<tr>
<td>Color</td>
<td><code>get_shadow_color () const</code></td>
</tr>
</tbody>
</table>

Continued on next page
Table 12.18 – continued from previous page

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>float</td>
<td>get_shadow_esm_multiplier() const</td>
</tr>
<tr>
<td>Object</td>
<td>get_texture() const</td>
</tr>
<tr>
<td>Vector2</td>
<td>get_texture_offset() const</td>
</tr>
<tr>
<td>float</td>
<td>get_texture_scale() const</td>
</tr>
<tr>
<td>int</td>
<td>get_z_range_max() const</td>
</tr>
<tr>
<td>int</td>
<td>get_z_range_min() const</td>
</tr>
<tr>
<td>bool</td>
<td>is_editor_only() const</td>
</tr>
<tr>
<td>bool</td>
<td>is_enabled() const</td>
</tr>
<tr>
<td>bool</td>
<td>is_shadow_enabled() const</td>
</tr>
<tr>
<td>void</td>
<td>set_color(Color color)</td>
</tr>
<tr>
<td>void</td>
<td>set_editor_only(bool editor_only)</td>
</tr>
<tr>
<td>void</td>
<td>set_enabled(bool enabled)</td>
</tr>
<tr>
<td>void</td>
<td>set_energy(float energy)</td>
</tr>
<tr>
<td>void</td>
<td>set_height(float height)</td>
</tr>
<tr>
<td>void</td>
<td>set_item_mask(int item_mask)</td>
</tr>
<tr>
<td>void</td>
<td>set_item_shadow_mask(int item_shadow_mask)</td>
</tr>
<tr>
<td>void</td>
<td>set_layer_range_max(int layer)</td>
</tr>
<tr>
<td>void</td>
<td>set_layer_range_min(int layer)</td>
</tr>
<tr>
<td>void</td>
<td>set_mode(int mode)</td>
</tr>
<tr>
<td>void</td>
<td>set_shadow_buffer_size(int size)</td>
</tr>
<tr>
<td>void</td>
<td>set_shadow_color(Color shadow_color)</td>
</tr>
<tr>
<td>void</td>
<td>set_shadow_enabled(bool enabled)</td>
</tr>
<tr>
<td>void</td>
<td>set_shadow_esm_multiplier(float multiplier)</td>
</tr>
<tr>
<td>void</td>
<td>set_texture(Object texture)</td>
</tr>
<tr>
<td>void</td>
<td>set_texture_offset(Vector2 texture_offset)</td>
</tr>
<tr>
<td>void</td>
<td>set_texture_scale(float texture_scale)</td>
</tr>
<tr>
<td>void</td>
<td>set_z_range_max(int z)</td>
</tr>
<tr>
<td>void</td>
<td>set_z_range_min(int z)</td>
</tr>
</tbody>
</table>

12.151.3 Numeric Constants

- **MODE_ADD = 0** — Adds the value of pixels corresponding to the Light2D to the values of pixels under it. This is the common behaviour of a light.

- **MODE_SUB = 1** — Subtract the value of pixels corresponding to the Light2D to the values of pixels under it, resulting in inverted light effect.

- **MODE_MIX = 2** — Mix the value of pixels corresponding to the Light2D to the values of pixels under it by linear interpolation.

- **MODE_MASK = 3** — The light texture of the Light2D is used as a mask, hiding or revealing parts of the screen underneath depending on the value of each pixel of the light (mask) texture.

12.151.4 Description

Node that casts light in a 2D environment. Light is defined by a (usually grayscale) texture, a color, an energy value, a mode (see constants), and various other parameters (range and shadows-related). Note that Light2D can be used as a mask.
12.151.5 Member Function Description

- **Color** get_color ( ) const
  Return the color of the Light2D.

- **float** get_energy ( ) const
  Return the energy value of the Light2D.

- **float** get_height ( ) const
  Return the height of the Light2D. Used with 2D normalmapping.

- **int** get_item_mask ( ) const
  Return the item mask of the Light2D.

- **int** get_item_shadow_mask ( ) const
  Return the item shadow mask of the Light2D.

- **int** get_layer_range_max ( ) const
  Set the maximum layer value of objects of the scene that are affected by the Light2D.

- **int** get_layer_range_min ( ) const
  Get the minimum layer value of objects of the scene that are affected by the Light2D.

- **int** get_mode ( ) const
  Return the current mode set to the Light2D.

- **int** get_shadow_buffer_size ( ) const
  Return the shadow buffer size.

- **Color** get_shadow_color ( ) const
  Return the color of casted shadows for this Light2D.

- **float** get_shadow_esm_multiplier ( ) const
  Return the Exponential Shadow Multiplier (ESM) value of the Light2D.

- **Object** get_texture ( ) const
  Return the texture of the Light2D.

- **Vector2** get_texture_offset ( ) const
  Return the offset of the light texture.

- **float** get_texture_scale ( ) const
  Return the scale value of the light texture.

- **int** get_z_range_max ( ) const
  Get the maximum Z value that objects of the scene can be in order to be affected by the Light2D.

- **int** get_z_range_min ( ) const
  Get the minimum Z value that objects of the scene have to be in order to be affected by the Light2D.

- **bool** is_editor_only ( ) const
  Return true if the Light2D is enabled, false if it is not.

- **bool** is_enabled ( ) const
  Return true if the Light2D is enabled, false if it is not.
• `bool is_shadow_enabled ( )` const

Return true if shadow casting is enabled for this Light2D, else return false.

• `void set_color ( Color color )`

Set the color of the Light2D.

• `void set_editor_only ( bool editor_only )`

• `void set_enabled ( bool enabled )`

Switches the Light2D on or off, depending on the ‘enabled’ parameter.

• `void set_energy ( float energy )`

Set the energy value of the Light2D. The bigger the value, the stronger the light.

• `void set_height ( float height )`

Set the height of the Light2D. Used with 2D normalmapping.

• `void set_item_mask ( int item_mask )`

Set the item mask of the Light2D to ‘item_mask’ value.

• `void set_item_shadow_mask ( int item_shadow_mask )`

Set the item shadow mask to ‘item_shadow_mask’ value.

• `void set_layer_range_max ( int layer )`

Set the maximum layer value of objects of the scene that are affected by the Light2D.

• `void set_layer_range_min ( int layer )`

Set the minimum layer value of objects of the scene that are affected by the Light2D.

• `void set_mode ( int mode )`

Set the behaviour mode of the Light2D. Use constants defined in the constants section.

• `void set_shadow_buffer_size ( int size )`

Set the shadow buffer size.

• `void set_shadow_color ( Color shadow_color )`

Set the color of casted shadows for this Light2D.

• `void set_shadow_enabled ( bool enabled )`

Enable or disable shadows casting from this Light2D according to the ‘enabled’ parameter.

• `void set_shadow_esm_multiplier ( float multiplier )`

Set the Exponential Shadow Multiplier (ESM) value of the Light2D.

• `void set_texture ( Object texture )`

Set the texture of the Light2D.

• `void set_texture_offset ( Vector2 texture_offset )`

Set the offset of the light texture.

• `void set_texture_scale ( float texture_scale )`

Set the scale value of the light texture.

• `void set_z_range_max ( int z )`
Set the maximum Z value that objects of the scene can be in order to be affected by the Light2D.

- void set_z_range_min ( int z )

Set the minimum Z value that objects of the scene have to be in order to be affected by the Light2D.

## 12.152 LightOccluder2D

**Inherits:** Node2D < CanvasItem < Node < Object  
**Category:** Core

### 12.152.1 Brief Description

Occludes light cast by a Light2D, thus casting shadows.

### 12.152.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>get_occluder_light_mask ( ) const</td>
</tr>
<tr>
<td>OccluderPolygon2D</td>
<td>get_occluder_polygon ( ) const</td>
</tr>
<tr>
<td>void</td>
<td>set_occluder_light_mask ( int mask )</td>
</tr>
<tr>
<td>void</td>
<td>set_occluder_polygon ( OccluderPolygon2D polygon )</td>
</tr>
</tbody>
</table>

### 12.152.3 Description

Occludes light cast by a Light2D, thus casting shadows. The LightOccluder2D must be provided with a shape (see OccluderPolygon2D) that allows the shadow to be computed. This shape affects the resulting shadow, while the shape of the representing asset shadowed does not actually affect shadows.

### 12.152.4 Member Function Description

- **int get_occluder_light_mask ( ) const**
  Return the light mask of the LightOccluder2D.

- **OccluderPolygon2D get_occluder_polygon ( ) const**
  Return the OccluderPolygon2D that defines the LightOccluder2D.

- **void set_occluder_light_mask ( int mask )**
  Set the LightOccluder2D light mask. The LightOccluder2D will cast shadows only from Light2Ds that belong to the same light mask(s).

- **void set_occluder_polygon ( OccluderPolygon2D polygon )**
  Set the OccluderPolygon2D that defines the LightOccluder2D.

## 12.153 LineEdit

**Inherits:** Control < CanvasItem < Node < Object  
**Category:** Core
12.153.1 Brief Description

Control that provides single line string editing.

12.153.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>append_at_cursor ( String text )</td>
</tr>
<tr>
<td>void</td>
<td>clear ()</td>
</tr>
<tr>
<td>bool</td>
<td>cursor_get_blink_enabled () const</td>
</tr>
<tr>
<td>float</td>
<td>cursor_get_blink_speed () const</td>
</tr>
<tr>
<td>void</td>
<td>cursor_set_blink_enabled ( bool enable )</td>
</tr>
<tr>
<td>void</td>
<td>cursor_set_blink_speed ( float blink_speed )</td>
</tr>
<tr>
<td>int</td>
<td>get_align () const</td>
</tr>
<tr>
<td>int</td>
<td>get_cursor_pos () const</td>
</tr>
<tr>
<td>int</td>
<td>get_max_length () const</td>
</tr>
<tr>
<td>PopupMenu</td>
<td>get_menu () const</td>
</tr>
<tr>
<td>String</td>
<td>get_placeholder () const</td>
</tr>
<tr>
<td>float</td>
<td>get_placeholder_alpha () const</td>
</tr>
<tr>
<td>String</td>
<td>get_text () const</td>
</tr>
<tr>
<td>bool</td>
<td>is_editable () const</td>
</tr>
<tr>
<td>bool</td>
<td>is_secret () const</td>
</tr>
<tr>
<td>void</td>
<td>menu_option ( int option )</td>
</tr>
<tr>
<td>void</td>
<td>select ( int from=0, int to=-1 )</td>
</tr>
<tr>
<td>void</td>
<td>select_all ()</td>
</tr>
<tr>
<td>void</td>
<td>set_align ( int align )</td>
</tr>
<tr>
<td>void</td>
<td>set_cursor_pos ( int pos )</td>
</tr>
<tr>
<td>void</td>
<td>set_editable ( bool enabled )</td>
</tr>
<tr>
<td>void</td>
<td>set_max_length ( int chars )</td>
</tr>
<tr>
<td>void</td>
<td>set_placeholder ( String text )</td>
</tr>
<tr>
<td>void</td>
<td>set_placeholder_alpha ( float alpha )</td>
</tr>
<tr>
<td>void</td>
<td>set_secret ( bool enabled )</td>
</tr>
<tr>
<td>void</td>
<td>set_text ( String text )</td>
</tr>
</tbody>
</table>

12.153.3 Signals

- text_changed ( String text )
  When the text changes, this signal is emitted.
- text_entered ( String text )
  This signal is emitted when the user presses KEY_ENTER on the LineEdit. This signal is often used as an alternate confirmation mechanism in dialogs.

12.153.4 Numeric Constants

- ALIGN_LEFT = 0
- ALIGN_CENTER = 1
- ALIGN_RIGHT = 2
- ALIGN_FILL = 3
Godot Engine Documentation, Release 2.1

- MENU_CUT = 0
- MENU_COPY = 1
- MENU_PASTE = 2
- MENU_CLEAR = 3
- MENU_SELECT_ALL = 4
- MENU_UNDO = 5
- MENU_MAX = 6

12.153.5 Description
LineEdit provides a single line string editor, used for text fields.

12.153.6 Member Function Description

- void append_at_cursor (String text)
Append text at cursor, scrolling the LineEdit when needed.
- void clear ()
Clear the LineEdit text.
- bool cursor_get_blink_enabled ( ) const
Gets whether the line edit caret is blinking.
- float cursor_get_blink_speed ( ) const
Gets the line edit caret blink speed.
- void cursor_set_blink_enabled (bool enable)
Set the line edit caret to blink.
- void cursor_set_blink_speed (float blink_speed)
Set the line edit caret blink speed. Cannot be less than or equal to 0.
- int get_align ( ) const
- int get_cursor_pos ( ) const
Return the cursor position inside the LineEdit.
- int get_max_length ( ) const
Return the maximum amount of characters the LineEdit can edit. If 0 is returned, no limit exists.
- PopupMenu get_menu ( ) const
- String get_placeholder ( ) const
- float get_placeholder_alpha ( ) const
- String get_text ( ) const
Return the text in the LineEdit.
- bool is_editable ( ) const
Return the editable status of the LineEdit (see set_editable).
• **bool is_secret ( ) const**

Return the secret status of the LineEdit (see set_secret).

• void **menu_option ( int option )**

• void **select ( int from=0, int to=-1 )**

• void **select_all ( )**

Select the whole string.

• void **set_align ( int align )**

• void **set_cursor_pos ( int pos )**

Set the cursor position inside the LineEdit, causing it to scroll if needed.

• void **set_editable ( bool enabled )**

Set the editable status of the LineEdit. When disabled, existing text can't be modified and new text can't be added.

• void **set_max_length ( int chars )**

Set the maximum amount of characters the LineEdit can edit, and cropping existing text in case it exceeds that limit. Setting 0 removes the limit.

• void **set_placeholder ( String text )**

• void **set_placeholder_alpha ( float alpha )**

• void **set_secret ( bool enabled )**

Set the secret status of the LineEdit. When enabled, every character is displayed as "*".

• void **set_text ( String text )**

Set the text in the LineEdit, clearing the existing one and the selection.

### 12.154 LineShape2D

**Inherits:** Shape2D < Resource < Reference < Object

**Category:** Core

### 12.154.1 Brief Description

Line shape for 2D collision objects.

### 12.154.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Member Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>float</td>
<td>get_d ( ) const</td>
</tr>
<tr>
<td>Vector2</td>
<td>get_normal ( ) const</td>
</tr>
<tr>
<td>void</td>
<td>set_d ( float d )</td>
</tr>
<tr>
<td>void</td>
<td>set_normal ( Vector2 normal )</td>
</tr>
</tbody>
</table>
12.154.3 Description

Line shape for 2D collision objects. It works like a 2D plane and will not allow any body to go to the negative side. Not recommended for rigid bodies, and usually not recommended for static bodies either because it forces checks against it on every frame.

12.154.4 Member Function Description

- float get_d () const
  Return the line distance from the origin.
  - Vector2 get_normal () const
  Return the line normal.
  - void set_d ( float d )
    Set the line distance from the origin.
  - void set_normal ( Vector2 normal )
    Set the line normal.

12.155 LinkButton

Inherits: BaseButton < Control < CanvasItem < Node < Object

Category: Core

12.155.1 Brief Description

Simple button used to represent a link to some resource

12.155.2 Member Functions

<table>
<thead>
<tr>
<th>String</th>
<th>get_text ( ) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>get_underline_mode ( ) const</td>
</tr>
<tr>
<td>void</td>
<td>set_text ( String text )</td>
</tr>
<tr>
<td>void</td>
<td>set_underline_mode ( int underline_mode )</td>
</tr>
</tbody>
</table>

12.155.3 Numeric Constants

- UNDERLINE_MODE_ALWAYS = 0 — The LinkButton will always show an underline at the bottom of its text
- UNDERLINE_MODE_ON_HOVER = 1 — The LinkButton will show an underline at the bottom of its text when the mouse cursor is over it.
- UNDERLINE_MODE.Never = 2 — The LinkButton will never show an underline at the bottom of its text.
12.155.4 Description

This kind of buttons are primarily used when the interaction with the button causes a context change (like linking to a web page).

12.155.5 Member Function Description

- **String get_text ( ) const**
  Returns the text of the button.
- **int get_underline_mode ( ) const**
  Returns the underline mode for this button.
- **void set_text ( String text )**
  Sets the text of the button.
- **void set_underline_mode ( int underline_mode )**
  Sets the underline mode for this button, the argument must be one of the `LinkButton` constants (see constants section).

12.156 Listener

**Inherits:** Spatial < Node < Object

**Category:** Core

12.156.1 Brief Description

12.156.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>clear_current ( )</td>
</tr>
<tr>
<td>Transform</td>
<td>get_listener_transform ( ) const</td>
</tr>
<tr>
<td>bool</td>
<td>is_current ( ) const</td>
</tr>
<tr>
<td>void</td>
<td>make_current ( )</td>
</tr>
</tbody>
</table>

12.156.3 Member Function Description

- **void clear_current ( )**
- **Transform get_listener_transform ( ) const**
- **bool is_current ( ) const**
- **void make_current ( )**

12.157 MainLoop

**Inherits:** Object

**Inherited By:** SceneTree
Category: Core

12.157.1 Brief Description

Main loop is the abstract main loop base class.

12.157.2 Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_drop_files</td>
<td>void(StringArray files, int screen) virtual</td>
</tr>
<tr>
<td>_finalize</td>
<td>void() virtual</td>
</tr>
<tr>
<td>_idle</td>
<td>void(float delta) virtual</td>
</tr>
<tr>
<td>_initialize</td>
<td>void() virtual</td>
</tr>
<tr>
<td>_input_event</td>
<td>void(InputEvent ev) virtual</td>
</tr>
<tr>
<td>_input_text</td>
<td>void(String text) virtual</td>
</tr>
<tr>
<td>_iteration</td>
<td>void(float delta) virtual</td>
</tr>
<tr>
<td>finish</td>
<td>void()</td>
</tr>
<tr>
<td>idle</td>
<td>bool(float delta)</td>
</tr>
<tr>
<td>init</td>
<td>void()</td>
</tr>
<tr>
<td>input_event</td>
<td>void(InputEvent ev) virtual</td>
</tr>
<tr>
<td>input_text</td>
<td>void(String text)</td>
</tr>
<tr>
<td>iteration</td>
<td>bool(float delta)</td>
</tr>
</tbody>
</table>

12.157.3 Numeric Constants

- NOTIFICATION_WM_MOUSE_ENTER = 3
- NOTIFICATION_WM_MOUSE_EXIT = 4
- NOTIFICATION_WM_FOCUS_IN = 5
- NOTIFICATION_WM_FOCUS_OUT = 6
- NOTIFICATION_WM_QUIT_REQUEST = 7
- NOTIFICATION_WM_UNFOCUS_REQUEST = 8
- NOTIFICATION_OS_MEMORY_WARNING = 9

12.157.4 Description

Main loop is the abstract main loop base class. All other main loop classes are derived from it. Upon application start, a MainLoop has to be provided to OS, else the application will exit. This happens automatically (and a SceneTree is created), unless a main Script is supplied, which may or not create and return a MainLoop.

12.157.5 Member Function Description

- void _drop_files(StringArray files, int screen) virtual
- void _finalize() virtual
- void _idle(float delta) virtual
- void _initialize() virtual
• void _input_event ( InputEvent ev ) virtual
• void _input_text ( String text ) virtual
• void _iteration ( float delta ) virtual
• void finish ()
• bool idle ( float delta )
• void init ()
• void input_event ( InputEvent ev )
• void input_text ( String text )
• bool iteration ( float delta )

12.158 MarginContainer

Inherits: Container < Control < CanvasItem < Node < Object
Category: Core

12.158.1 Brief Description

Simple margin container.

12.158.2 Description

Simple margin container. Adds custom margins to anything contained.

12.159 Marshalls

Inherits: Reference < Object
Category: Core

12.159.1 Brief Description

12.159.2 Member Functions

<table>
<thead>
<tr>
<th>RawArray</th>
<th>base64_to_raw ( String base64_str )</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>base64_to_utf8 ( String base64_str )</td>
</tr>
<tr>
<td>Variant</td>
<td>base64_to_variant ( String base64_str )</td>
</tr>
<tr>
<td>String</td>
<td>raw_to_base64 ( RawArray array )</td>
</tr>
<tr>
<td>String</td>
<td>utf8_to_base64 ( String utf8_str )</td>
</tr>
<tr>
<td>String</td>
<td>variant_to_base64 ( Variant variant )</td>
</tr>
</tbody>
</table>
12.159.3 Member Function Description

- `RawArray base64_to_raw ( String base64_str )`
- `String base64_to_utf8 ( String base64_str )`
- `Variant base64_to_variant ( String base64_str )`
- `String raw_to_base64 ( RawArray array )`
- `String utf8_to_base64 ( String utf8_str )`
- `String variant_to_base64 ( Variant variant )`

12.160 Material

Inherits: `Resource < Reference < Object`

Inherited By: `ShaderMaterial, FixedMaterial`

Category: Core

12.160.1 Brief Description

Abstract base `Resource` for coloring and shading geometry.

12.160.2 Member Functions

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>int</code></td>
<td><code>get_blend_mode ( ) const</code></td>
</tr>
<tr>
<td><code>int</code></td>
<td><code>get_depth_draw_mode ( ) const</code></td>
</tr>
<tr>
<td><code>bool</code></td>
<td><code>get_flag ( int flag ) const</code></td>
</tr>
<tr>
<td><code>float</code></td>
<td><code>get_line_width ( ) const</code></td>
</tr>
<tr>
<td><code>void</code></td>
<td><code>set_blend_mode ( int mode )</code></td>
</tr>
<tr>
<td><code>void</code></td>
<td><code>set_depth_draw_mode ( int mode )</code></td>
</tr>
<tr>
<td><code>void</code></td>
<td><code>set_flag ( int flag, bool enable )</code></td>
</tr>
<tr>
<td><code>void</code></td>
<td><code>set_line_width ( float width )</code></td>
</tr>
</tbody>
</table>

12.160.3 Numeric Constants

- `FLAG_VISIBLE = 0` — Geometry is visible when this flag is enabled (default).
- `FLAG_DOUBLE_SIDED = 1` — Both front facing and back facing triangles are rendered when this flag is enabled.
- `FLAG_INVERT_FACES = 2` — Front facing and back facing order is swapped when this flag is enabled.
- `FLAG_UNSHADED = 3` — Shading (lighting) is disabled when this flag is enabled.
- `FLAG_ONTOP = 4`
- `FLAG_LIGHTMAP_ON_UV2 = 5`
- `FLAG_COLOR_ARRAY_SRGB = 6`
- `FLAG_MAX = 7` — Maximum amount of flags.
- `DEPTH_DRAW_ALWAYS = 0`
• DEPTH_DRAW_OPAQUE_ONLY = 1
• DEPTH_DRAW_OPAQUE_PRE_PASS_ALPHA = 2
• DEPTH_DRAW_NEVER = 3
• BLEND_MODE_MIX = 0 — Use the regular alpha blending equation (source and dest colors are faded) (default).
• BLEND_MODE_ADD = 1 — Use additive blending equation, often used for particle effects such as fire or light decals.
• BLEND_MODE_SUB = 2 — Use subtractive blending equation, often used for some smoke effects or types of glass.
• BLEND_MODE_MUL = 3
• BLEND_MODE_PREMULT_ALPHA = 4

12.160.4 Description

Material is a base Resource used for coloring and shading geometry. All materials inherit from it and almost all VisualInstance derived nodes carry a Material. A few flags and parameters are shared between all material types and are configured here.

12.160.5 Member Function Description

• int get_blend_mode ( ) const
  Return blend mode for the material, which can be one of BLEND_MODE_MIX (default), BLEND_MODE_ADD, BLEND_MODE_SUB. Keep in mind that only BLEND_MODE_MIX ensures that the material may be opaque, any other blend mode will render with alpha blending enabled in raster-based VisualServer implementations.

• int get_depth_draw_mode ( ) const

• bool get_flag ( int flag ) const
  Return a Material flag, which toggles on or off a behavior when rendering. See enumeration FLAG_* for a list.

• float get_line_width ( ) const
  Return the line width for geometry drawn with FLAG_WIREFRAME enabled, or LINE primitives. Note that not all hardware or VisualServer backends support this (like DirectX).

• void set_blend_mode ( int mode )
  Set blend mode for the material, which can be one of BLEND_MODE_MIX (default), BLEND_MODE_ADD, BLEND_MODE_SUB. Keep in mind that only BLEND_MODE_MIX ensures that the material may be opaque, any other blend mode will render with alpha blending enabled in raster-based VisualServer implementations.

• void set_depth_draw_mode ( int mode )

• void set_flag ( int flag, bool enable )
  Set a Material flag, which toggles on or off a behavior when rendering. See enumeration FLAG_* for a list.

• void set_line_width ( float width )
  Set the line width for geometry drawn with FLAG_WIREFRAME enabled, or LINE primitives. Note that not all hardware or VisualServer backends support this (like DirectX).
12.161 MaterialShader

Inherits: Shader < Resource < Reference < Object
Category: Core

12.161.1 Brief Description

12.162 MaterialShaderGraph

Inherits: ShaderGraph < Shader < Resource < Reference < Object
Category: Core

12.162.1 Brief Description

12.163 Matrix3

Category: Built-In Types

12.163.1 Brief Description

3x3 matrix datatype.

12.163.2 Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matrix3 ( Quat from )</td>
<td></td>
</tr>
<tr>
<td>Matrix3 ( Vector3 axis, float phi )</td>
<td></td>
</tr>
<tr>
<td>Matrix3 ( Vector3 x_axis, Vector3 y_axis, Vector3 z_axis )</td>
<td></td>
</tr>
<tr>
<td>float determinant ( )</td>
<td></td>
</tr>
<tr>
<td>Vector3 get_euler ( )</td>
<td></td>
</tr>
<tr>
<td>int get_orthogonal_index ( )</td>
<td></td>
</tr>
<tr>
<td>Vector3 get_scale ( )</td>
<td></td>
</tr>
<tr>
<td>Matrix3 inverse ( )</td>
<td></td>
</tr>
<tr>
<td>Matrix3 orthonormalized ( )</td>
<td></td>
</tr>
<tr>
<td>Matrix3 rotated ( Vector3 axis, float phi )</td>
<td></td>
</tr>
<tr>
<td>Matrix3 scaled ( Vector3 scale )</td>
<td></td>
</tr>
<tr>
<td>float tdotx ( Vector3 with )</td>
<td></td>
</tr>
<tr>
<td>float tdoty ( Vector3 with )</td>
<td></td>
</tr>
<tr>
<td>float tdotz ( Vector3 with )</td>
<td></td>
</tr>
<tr>
<td>Matrix3 transposed ( )</td>
<td></td>
</tr>
<tr>
<td>Vector3 xform ( Vector3 v )</td>
<td></td>
</tr>
<tr>
<td>Vector3 xform_inv ( Vector3 v )</td>
<td></td>
</tr>
</tbody>
</table>

12.163.3 Member Variables

- Vector3 x
• Vector3 y
• Vector3 z

12.163.4 Description

3x3 matrix used for 3D rotation and scale. Contains 3 vector fields x, y and z. Can also be accessed as array of 3D vectors. Almost always used as orthogonal basis for a Transform.

12.163.5 Member Function Description

• Matrix3 Matrix3 ( Quat from )
  Create a matrix from a quaternion.

• Matrix3 Matrix3 ( Vector3 axis, float phi )
  Create a matrix from an axis vector and an angle.

• Matrix3 Matrix3 ( Vector3 x_axis, Vector3 y_axis, Vector3 z_axis )
  Create a matrix from 3 axis vectors.

• float determinant ( )
  Return the determinant of the matrix.

• Vector3 get_euler ( )
  Return euler angles from the matrix.

• int get_orthogonal_index ( )
• Vector3 get_scale ( )
• Matrix3 inverse ( )
  Return the affine inverse of the matrix.

• Matrix3 orthonormalized ( )
  Return the orthonormalized version of the matrix (useful to call from time to time to avoid rounding error).

• Matrix3 rotated ( Vector3 axis, float phi )
  Return the rotated version of the matrix, by a given axis and angle.

• Matrix3 scaled ( Vector3 scale )
  Return the scaled version of the matrix, by a 3D scale.

• float tdotx ( Vector3 with )
  Transposed dot product with the x axis of the matrix.

• float tdoty ( Vector3 with )
  Transposed dot product with the y axis of the matrix.

• float tdotz ( Vector3 with )
  Transposed dot product with the z axis of the matrix.

• Matrix3 transposed ( )
  Return the transposed version of the matrix.
• \texttt{Vector3 xform ( Vector3 v )}

Return a vector transformed by the matrix and return it.

• \texttt{Vector3 xform\_inv ( Vector3 v )}

Return a vector transformed by the transposed matrix and return it.

12.164 Matrix32

Category: Built-In Types

12.164.1 Brief Description

3x2 Matrix for 2D transforms.

12.164.2 Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{Matrix32 ( Transform from )}</td>
<td></td>
</tr>
<tr>
<td>\texttt{Matrix32 ( Vector2 x_axis, Vector2 y_axis, Vector2 origin )}</td>
<td></td>
</tr>
<tr>
<td>\texttt{Matrix32 ( float rot, Vector2 pos )}</td>
<td></td>
</tr>
<tr>
<td>\texttt{affine_inverse ( )}</td>
<td></td>
</tr>
<tr>
<td>\texttt{basis_xform ( var v )}</td>
<td></td>
</tr>
<tr>
<td>\texttt{basis_xform_inv ( var v )}</td>
<td></td>
</tr>
<tr>
<td>\texttt{get_origin ( )}</td>
<td></td>
</tr>
<tr>
<td>\texttt{get_rotation ( )}</td>
<td></td>
</tr>
<tr>
<td>\texttt{get_scale ( )}</td>
<td></td>
</tr>
<tr>
<td>\texttt{interpolate_with ( Matrix32 m, float c )}</td>
<td></td>
</tr>
<tr>
<td>\texttt{inverse ( )}</td>
<td></td>
</tr>
<tr>
<td>\texttt{orthonormalized ( )}</td>
<td></td>
</tr>
<tr>
<td>\texttt{rotated ( float phi )}</td>
<td></td>
</tr>
<tr>
<td>\texttt{scaled ( Vector2 scale )}</td>
<td></td>
</tr>
<tr>
<td>\texttt{translated ( Vector2 offset )}</td>
<td></td>
</tr>
<tr>
<td>\texttt{xform ( var v )}</td>
<td></td>
</tr>
<tr>
<td>\texttt{xform_inv ( var v )}</td>
<td></td>
</tr>
</tbody>
</table>

12.164.3 Member Variables

• \texttt{Vector2 o}
• \texttt{Vector2 x}
• \texttt{Vector2 y}

12.164.4 Description

3x2 Matrix for 2D transforms.
12.164.5 Member Function Description

- `Matrix32 Matrix32 (Transform from)`
- `Matrix32 Matrix32 (Vector2 x_axis, Vector2 y_axis, Vector2 origin)`
- `Matrix32 Matrix32 (float rot, Vector2 pos)`
- `Matrix32 affine_inverse ()`
- `Matrix32 basis_xform (var v)`
- `Matrix32 basis_xform_inv (var v)`
- `Vector2 get_origin ()`
- `float get_rotation ()`
- `Vector2 get_scale ()`
- `Matrix32 interpolate_with (Matrix32 m, float c)`
- `Matrix32 inverse ()`
- `Matrix32 orthonormalized ()`
- `Matrix32 rotated (float phi)`
- `Matrix32 scaled (Vector2 scale)`
- `Matrix32 translated (Vector2 offset)`
- `Matrix32 xform (var v)`
- `Matrix32 xform_inv (var v)`

12.165 MenuButton

Inherits: `Button < BaseButton < Control < CanvasItem < Node < Object`
Category: Core

12.165.1 Brief Description

Special button that brings up a `PopupMenu` when clicked.

12.165.2 Member Functions

```
PopupMenu get_popup ()
```

12.165.3 Signals

- `about_to_show ()`
12.165.4 Description

Special button that brings up a `PopupMenu` when clicked. That’s pretty much all it does, as it’s just a helper class when building GUIs.

12.165.5 Member Function Description

- `PopupMenu get_popup ( )`

Return the `PopupMenu` contained in this button.

12.166 Mesh

Inherits: `Resource < Reference < Object`

Category: Core

12.166.1 Brief Description

A `Resource` that contains vertex-array based geometry.

12.166.2 Member Functions

```cpp
void add_morph_target ( String name )
void add_surface ( int primitive, Array arrays, Array morph_arrays=Array(), bool alphasort=false )
void center_geometry ( )
void clear_morph_targets ( )
Shape create_convex_shape ( ) const
Mesh create_outline ( float margin ) const
Shape create_trimesh_shape ( ) const
TriangleMesh generate_triangle_mesh ( ) const
AABB get_custom_aabb ( ) const
Vector3Array get_faces ( ) const
int get_morph_target_count ( ) const
int get_morph_target_mode ( ) const
String get_morph_target_name ( int index ) const
int get_surface_count ( ) const
void regen_normalmaps ( )
void set_custom_aabb ( AABB aabb )
void set_morph_target_mode ( int mode )
int surface_get_array_index_len ( int surf_idx ) const
int surface_get_array_len ( int surf_idx ) const
int surface_get_format ( int surf_idx ) const
Material surface_get_material ( int surf_idx ) const
String surface_get_name ( int surf_idx ) const
int surface_get_primitive_type ( int surf_idx ) const
void surface_remove ( int surf_idx )
void surface_set_material ( int surf_idx, Material material )
void surface_set_name ( int surf_idx, String name )
```
12.166.3 Numeric Constants

- **NO_INDEX_ARRAY** = -1 — Default value used for index_array_len when no indices are present.
- **ARRAY_WEIGHTS_SIZE** = 4 — Amount of weights/bone indices per vertex (always 4).
- **ARRAY_VERTEX** = 0 — Vertex array (array of Vector3 vertices).
- **ARRAY_NORMAL** = 1 — Normal array (array of Vector3 normals).
- **ARRAY_TANGENT** = 2 — Tangent array, array of groups of 4 floats. first 3 floats determine the tangent, and the last the binormal direction as -1 or 1.
- **ARRAY_COLOR** = 3 — Vertex array (array of Color colors).
- **ARRAY_TEX_UV** = 4 — UV array (array of Vector3 UVs or float array of groups of 2 floats (u,v)).
- **ARRAY_TEX_UV2** = 5 — Second UV array (array of Vector3 UVs or float array of groups of 2 floats (u,v)).
- **ARRAY_BONES** = 6 — Array of bone indices, as a float array. Each element in groups of 4 floats.
- **ARRAY_WEIGHTS** = 7 — Array of bone weights, as a float array. Each element in groups of 4 floats.
- **ARRAY_INDEX** = 8 — Array of integers, used as indices referencing vertices. No index can be beyond the vertex array size.
- **ARRAY_FORMAT_VERTEX** = 1 — Array format will include vertices (mandatory).
- **ARRAY_FORMAT_NORMAL** = 2 — Array format will include normals
- **ARRAY_FORMAT_TANGENT** = 4 — Array format will include tangents
- **ARRAY_FORMAT_COLOR** = 8 — Array format will include a color array.
- **ARRAY_FORMAT_TEX_UV** = 16 — Array format will include UVs.
- **ARRAY_FORMAT_TEX_UV2** = 32 — Array format will include another set of UVs.
- **ARRAY_FORMAT_BONES** = 64 — Array format will include bone indices.
- **ARRAY_FORMAT_WEIGHTS** = 128 — Array format will include bone weights.
- **ARRAY_FORMAT_INDEX** = 256 — Index array will be used.
- **PRIMITIVE_POINTS** = 0 — Render array as points (one vertex equals one point).
- **PRIMITIVE_LINES** = 1 — Render array as lines (every two vertices a line is created).
- **PRIMITIVE_LINE_STRIP** = 2 — Render array as line strip.
- **PRIMITIVE_LINE_LOOP** = 3 — Render array as line loop (like line strip, but closed).
- **PRIMITIVE_TRIANGLES** = 4 — Render array as triangles (every three vertices a triangle is created).
- **PRIMITIVE_TRIANGLE_STRIP** = 5 — Render array as triangle strips.
- **PRIMITIVE_TRIANGLE_FAN** = 6 — Render array as triangle fans.

12.166.4 Description

Mesh is a type of Resource that contains vertex-array based geometry, divided in surfaces. Each surface contains a completely separate array and a material used to draw it. Design wise, a mesh with multiple surfaces is preferred to a single surface, because objects created in 3D editing software commonly contain multiple materials.
12.166.5 Member Function Description

- **void** add_morph_target ( *String* name )
- **void** add_surface ( *int* primitive, *Array* arrays, *Array* morph_arrays=Array(), *bool* alphasort=false )

Create a new surface (*get_surface_count*) that will become surf_idx for this.

Surfaces are created to be rendered using a “primitive”, which may be PRIMITIVE_POINTS, PRIMITIVE_LINES, PRIMITIVE_LINE_STRIP, PRIMITIVE_LINE_LOOP, PRIMITIVE_TRIANGLES, PRIMITIVE_TRIANGLE_STRIP, PRIMITIVE_TRIANGLE_FAN. (As a note, when using indices, it is recommended to only use just points, lines or triangles).

The format of a surface determines which arrays it will allocate and hold, so “format” is a combination of ARRAY_FORMAT_* mask constants ORed together. ARRAY_FORMAT_VERTEX must be always present. “array_len” determines the amount of vertices in the array (not primitives!). if ARRAY_FORMAT_INDEX is in the format mask, then it means that an index array will be allocated and “index_array_len” must be passed.

- **void** center_geometry ( )
- **void** clear_morph_targets ( )
- *Shape* create_convex_shape ( ) const
- *Mesh* create_outline ( *float* margin ) const
- *Shape* create_trimesh_shape ( ) const
- *TriangleMesh* generate_triangle_mesh ( ) const
- *AABB* get_custom_aabb ( ) const
- *Vector3Array* get_faces ( ) const
- *int* get_morph_target_count ( ) const
- *int* get_morph_target_mode ( ) const
- *String* get_morph_target_name ( *int* index ) const
- *int* get_surface_count ( ) const

Return the amount of surfaces that the *Mesh* holds.

- **void** regen_normalmaps ( )
- **void** set_custom_aabb ( *AABB* aabb )
- **void** set_morph_target_mode ( *int* mode )
- *int* surface_get_array_index_len ( *int* surf_idx ) const

Return the length in indices of the index array in the requested surface (see *add_surface*).

- *int* surface_get_array_len ( *int* surf_idx ) const

Return the length in vertices of the vertex array in the requested surface (see *add_surface*).

- *int* surface_get_format ( *int* surf_idx ) const

Return the format mask of the requested surface (see *add_surface*).

- *Material* surface_get_material ( *int* surf_idx ) const

Return a *Material* in a given surface. Surface is rendered using this material.

- *String* surface_get_name ( *int* surf_idx ) const
- *int* surface_get_primitive_type ( *int* surf_idx ) const
Return the primitive type of the requested surface (see add_surface).

- void **surface_remove** ( int surf_idx )

Remove a surface at position surf_idx, shifting greater surfaces one surf_idx slot down.

- void **surface_set_material** ( int surf_idx, Material material )

Set a Material for a given surface. Surface will be rendered using this material.

- void **surface_set_name** ( int surf_idx, String name )

---

### 12.167 MeshDataTool

**Inherits:** Reference &lt; Object  
**Category:** Core

#### 12.167.1 Brief Description

#### 12.167.2 Member Functions

<table>
<thead>
<tr>
<th>Member Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>void clear ()</td>
</tr>
<tr>
<td>int commit_to_surface ( Object mesh )</td>
</tr>
<tr>
<td>int create_from_surface ( Object mesh, int surface )</td>
</tr>
<tr>
<td>int get_edge_count () const</td>
</tr>
<tr>
<td>IntArray get_edge_faces ( int idx ) const</td>
</tr>
<tr>
<td>void get_edge_meta ( int idx ) const</td>
</tr>
<tr>
<td>int get_edge_vertex ( int idx, int vertex ) const</td>
</tr>
<tr>
<td>int get_face_count () const</td>
</tr>
<tr>
<td>int get_face_edge ( int idx, int edge ) const</td>
</tr>
<tr>
<td>void get_face_meta ( int idx ) const</td>
</tr>
<tr>
<td>Vector3 get_face_normal ( int idx ) const</td>
</tr>
<tr>
<td>int get_face_vertex ( int idx, int vertex ) const</td>
</tr>
<tr>
<td>int get_format () const</td>
</tr>
<tr>
<td>Object get_material () const</td>
</tr>
<tr>
<td>Vector3 get_vertex ( int idx ) const</td>
</tr>
<tr>
<td>IntArray get_vertex_bones ( int idx ) const</td>
</tr>
<tr>
<td>Color get_vertex_color ( int idx ) const</td>
</tr>
<tr>
<td>int get_vertex_count () const</td>
</tr>
<tr>
<td>IntArray get_vertex_edges ( int idx ) const</td>
</tr>
<tr>
<td>IntArray get_vertex_faces ( int idx ) const</td>
</tr>
<tr>
<td>void get_vertex_meta ( int idx ) const</td>
</tr>
<tr>
<td>Vector3 get_vertex_normal ( int idx ) const</td>
</tr>
<tr>
<td>Plane get_vertex_tangent ( int idx ) const</td>
</tr>
<tr>
<td>Vector2 get_vertex_uv ( int idx ) const</td>
</tr>
<tr>
<td>Vector2 get_vertex_uv2 ( int idx ) const</td>
</tr>
<tr>
<td>RealArray get_vertex_weights ( int idx ) const</td>
</tr>
<tr>
<td>void set_edge_meta ( int idx, Variant meta )</td>
</tr>
<tr>
<td>void set_face_meta ( int idx, Variant meta )</td>
</tr>
<tr>
<td>void set_material ( Material material )</td>
</tr>
</tbody>
</table>
Table 12.19 – continued from previous page

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void set_vertex (int idx, Vector3 vertex)</td>
<td>Set vertex at index <code>idx</code> with <code>vertex</code> values</td>
</tr>
<tr>
<td>void set_vertex_bones (int idx, IntArray bones)</td>
<td>Set vertex bones at index <code>idx</code> with <code>bones</code> array</td>
</tr>
<tr>
<td>void set_vertex_color (int idx, Color color)</td>
<td>Set vertex color at index <code>idx</code> with <code>color</code> values</td>
</tr>
<tr>
<td>void set_vertex_meta (int idx, Variant meta)</td>
<td>Set vertex meta at index <code>idx</code> with <code>meta</code> value</td>
</tr>
<tr>
<td>void set_vertex_normal (int idx, Vector3 normal)</td>
<td>Set vertex normal at index <code>idx</code> with <code>normal</code> values</td>
</tr>
<tr>
<td>void set_vertex_tangent (int idx, Plane tangent)</td>
<td>Set vertex tangent at index <code>idx</code> with <code>tangent</code> values</td>
</tr>
<tr>
<td>void set_vertex_uv (int idx, Vector2 uv)</td>
<td>Set vertex UV at index <code>idx</code> with <code>uv</code> values</td>
</tr>
<tr>
<td>void set_vertex_uv2 (int idx, Vector2 uv2)</td>
<td>Set vertex UV2 at index <code>idx</code> with <code>uv2</code> values</td>
</tr>
<tr>
<td>void set_vertex_weights (int idx, RealArray weights)</td>
<td>Set vertex weights at index <code>idx</code> with <code>weights</code> array</td>
</tr>
</tbody>
</table>

### 12.167.3 Member Function Description

- void clear ( )
- int commit_to_surface (Object mesh)
- int create_from_surface (Object mesh, int surface)
- int get_edge_count ( ) const
- IntArray get_edge_faces (int idx) const
- void get_edge_meta (int idx) const
- int get_edge_vertex (int idx, int vertex) const
- int get_face_count ( ) const
- int get_face_edge (int idx, int edge) const
- void get_face_meta (int idx) const
- Vector3 get_face_normal (int idx) const
- int get_face_vertex (int idx, int vertex) const
- int get_format ( ) const
- Object get_material ( ) const
- Vector3 get_vertex (int idx) const
- IntArray get_vertex_bones (int idx) const
- Color get_vertex_color (int idx) const
- int get_vertex_count ( ) const
- IntArray get_vertex_edges (int idx) const
- IntArray get_vertex_faces (int idx) const
- void get_vertex_meta (int idx) const
- Vector3 get_vertex_normal (int idx) const
- Plane get_vertex_tangent (int idx) const
- Vector2 get_vertex_uv (int idx) const
- Vector2 get_vertex_uv2 (int idx) const
- RealArray get_vertex_weights (int idx) const
- void set_edge_meta (int idx, Variant meta)
• void set_face_meta ( int idx, Variant meta )
• void set_material ( Material material )
• void set_vertex ( int idx, Vector3 vertex )
• void set_vertex_bones ( int idx, IntArray bones )
• void set_vertex_color ( int idx, Color color )
• void set_vertex_meta ( int idx, Variant meta )
• void set_vertex_normal ( int idx, Vector3 normal )
• void set_vertex_tangent ( int idx, Plane tangent )
• void set_vertex_uv ( int idx, Vector2 uv )
• void set_vertex_uv2 ( int idx, Vector2 uv2 )
• void set_vertex_weights ( int idx, RealArray weights )

12.168 MeshInstance

Inherits: GeometryInstance < VisualInstance < Spatial < Node < Object

Category: Core

12.168.1 Brief Description

Node that instances meshes into a scenario.

12.168.2 Member Functions

| void | create_convex_collision ( ) |
| void | create_trimesh_collision ( ) |
| AABB | get_aabb ( ) const |
| Mesh | get_mesh ( ) const |
| NodePath | get_skeleton_path ( ) |
| void | set_mesh ( Mesh mesh ) |
| void | set_skeleton_path ( NodePath skeleton_path ) |

12.168.3 Description

MeshInstance is a Node that takes a Mesh resource and adds it to the current scenario by creating an instance of it. This is the class most often used to get 3D geometry rendered and can be used to instance a single Mesh in many places. This allows to reuse geometry and save on resources. When a Mesh has to be instanced more than thousands of times at close proximity, consider using a MultiMesh in a MultiMeshInstance instead.

12.168.4 Member Function Description

• void create_convex_collision ( )
• void create_trimesh_collision ( )
This helper creates a StaticBody child Node using the mesh geometry as collision. It’s mainly used for testing.

- **AABB** `get_aabb ( ) const`
  
  Return the AABB of the mesh, in local coordinates.

- **Mesh** `get_mesh ( ) const`
  
  Return the current Mesh resource for the instance.

- **NodePath** `get_skeleton_path ( )`

- void `set_mesh ( Mesh mesh )`
  
  Set the Mesh resource for the instance.

- void `set_skeleton_path ( NodePath skeleton_path )`

### 12.169 MeshLibrary

**Inherits:** Resource < Reference < Object

**Category:** Core

#### 12.169.1 Brief Description

Library of meshes.

#### 12.169.2 Member Functions

<table>
<thead>
<tr>
<th>void</th>
<th>clear ( )</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>create_item ( int id )</td>
</tr>
<tr>
<td>IntArray</td>
<td>get_item_list ( ) const</td>
</tr>
<tr>
<td>Mesh</td>
<td>get_item_mesh ( int id ) const</td>
</tr>
<tr>
<td>String</td>
<td>get_item_name ( int id ) const</td>
</tr>
<tr>
<td>NavigationMesh</td>
<td>get_item_navmesh ( int id ) const</td>
</tr>
<tr>
<td>Shape</td>
<td>get_item_shape ( int id ) const</td>
</tr>
<tr>
<td>int</td>
<td>get_last_unused_item_id ( ) const</td>
</tr>
<tr>
<td>void</td>
<td>remove_item ( int id )</td>
</tr>
<tr>
<td>void</td>
<td>set_item_mesh ( int id, Mesh mesh )</td>
</tr>
<tr>
<td>void</td>
<td>set_item_name ( int id, String name )</td>
</tr>
<tr>
<td>void</td>
<td>set_item_navmesh ( int id, NavigationMesh navmesh )</td>
</tr>
<tr>
<td>void</td>
<td>set_item_shape ( int id, Shape shape )</td>
</tr>
</tbody>
</table>

#### 12.169.3 Description

Library of meshes. Contains a list of Mesh resources, each with name and ID. Useful for GridMap or painting Terrain.

#### 12.169.4 Member Function Description

- void `clear ( )`

  Clear the library.
• void create_item ( int id )
Create a new item in the library, supplied an id.

• IntArray get_item_list ( ) const
Return the list of items.

• Mesh get_item_mesh ( int id ) const
Return the mesh of the item.

• String get_item_name ( int id ) const
Return the name of the item.

• NavigationMesh get_item_navmesh ( int id ) const
• Shape get_item_shape ( int id ) const
• int get_last_unused_item_id ( ) const
Get an unused id for a new item.

• void remove_item ( int id )
Remove the item.

• void set_item_mesh ( int id, Mesh mesh )
Set the mesh of the item.

• void set_item_name ( int id, String name )
Set the name of the item.

• void set_item_navmesh ( int id, NavigationMesh navmesh )
• void set_item_shape ( int id, Shape shape )

12.170 MultiMesh

Inherits: Resource < Reference < Object

Category: Core

12.170.1 Brief Description

Provides high performance mesh instancing.
## 12.170.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td><code>generate_aabb()</code></td>
</tr>
<tr>
<td>AABB</td>
<td><code>get_aabb()</code></td>
</tr>
<tr>
<td>Color</td>
<td><code>get_instance_color(int instance) const</code></td>
</tr>
<tr>
<td>int</td>
<td><code>get_instance_count()</code> const</td>
</tr>
<tr>
<td>Transform</td>
<td><code>get_instance_transform(int instance) const</code></td>
</tr>
<tr>
<td>Mesh</td>
<td><code>get_mesh()</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_aabb(AABB visibility_aabb)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_instance_color(int instance, Color color)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_instance_count(int count)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_instance_transform(int instance, Transform transform)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_mesh(Mesh mesh)</code></td>
</tr>
</tbody>
</table>

## 12.170.3 Description

MultiMesh provides low level mesh instancing. If the amount of Mesh instances needed goes from hundreds to thousands (and most need to be visible at close proximity) creating such a large amount of MeshInstance nodes may affect performance by using too much CPU or video memory.

For this case a MultiMesh becomes very useful, as it can draw thousands of instances with little API overhead.

As a drawback, if the instances are too far away of each other, performance may be reduced as every single instance will always rendered (they are spatially indexed as one, for the whole object).

Since instances may have any behavior, the AABB used for visibility must be provided by the user, or generated with `generate_aabb`.

## 12.170.4 Member Function Description

- **void** `generate_aabb()`
  
  Generate a new visibility AABB, using mesh AABB and instance transforms. Since instance information is stored in the VisualServer, this function is VERY SLOW and must NOT be used often.

- **AABB** `get_aabb()` const
  
  Return the visibility AABB.

- **Color** `get_instance_color(int instance) const`
  
  Get the color of a specific instance.

- **int** `get_instance_count()` const
  
  Return the amount of instances that is going to be drawn.

- **Transform** `get_instance_transform(int instance) const`
  
  Return the transform of a specific instance.

- **Mesh** `get_mesh()` const
  
  Return the Mesh resource drawn as multiple instances.

- **void** `set_aabb(AABB visibility_aabb)`
  
  Set the visibility AABB. If not provided, MultiMesh will not be visible.

- **void** `set_instance_color(int instance, Color color)`

---

**12.170. MultiMesh**
Set the color of a specific instance.

- void set_instance_count ( int count )

Set the amount of instances that is going to be drawn. Changing this number will erase all the existing instance transform and color data.

- void set_instance_transform ( int instance, Transform transform )

Set the transform for a specific instance.

- void set_mesh ( Mesh mesh )

Set the Mesh resource to be drawn in multiple instances.

12.171 MultiMeshInstance

Inherits: GeometryInstance < VisualInstance < Spatial < Node < Object

Category: Core

12.171.1 Brief Description

Node that instances a MultiMesh.

12.171.2 Member Functions

<table>
<thead>
<tr>
<th>Object</th>
<th>get_multimesh ( ) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>set_multimesh ( Object multimesh )</td>
</tr>
</tbody>
</table>

12.171.3 Description

MultiMeshInstance is a Node that takes a MultiMesh resource and adds it to the current scenario by creating an instance of it (yes, this is an instance of instances).

12.171.4 Member Function Description

- Object get_multimesh ( ) const

Return the MultiMesh that is used for instancing.

- void set_multimesh ( Object multimesh )

Set the MultiMesh to be instance.

12.172 Mutex

Inherits: Reference < Object

Category: Core
12.172.1 Brief Description

A synchronization Mutex.

12.172.2 Member Functions

<table>
<thead>
<tr>
<th>void</th>
<th>lock ( )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error</td>
<td>try_lock ( )</td>
</tr>
<tr>
<td>void</td>
<td>unlock ( )</td>
</tr>
</tbody>
</table>

12.172.3 Description

A synchronization Mutex. Element used in multi-threading. Basically a binary Semaphore. Guarantees that only one thread has this lock, can be used to protect a critical section.

12.172.4 Member Function Description

- void lock ( )
  Lock this Mutex, blocks until it is unlocked by the current owner.

- Error try_lock ( )
  Try locking this Mutex, does not block. Returns OK on success else ERR_BUSY.

- void unlock ( )
  Unlock this Mutex, leaving it to others threads.

12.173 Navigation

Inherits: Spatial < Node < Object
Category: Core

12.173.1 Brief Description

12.173.2 Member Functions

<table>
<thead>
<tr>
<th>Vector3</th>
<th>get_closest_point ( Vector3 to_point )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vector3</td>
<td>get_closest_point_normal ( Vector3 to_point )</td>
</tr>
<tr>
<td>Object</td>
<td>get_closest_point_owner ( Vector3 to_point )</td>
</tr>
<tr>
<td>Vector3</td>
<td>get_closest_point_to_segment ( Vector3 start, Vector3 end, bool use_collision=false )</td>
</tr>
<tr>
<td>Vector3Array</td>
<td>get_simple_path ( Vector3 start, Vector3 end, bool optimize=true )</td>
</tr>
<tr>
<td>Vector3</td>
<td>get_up_vector ( ) const</td>
</tr>
<tr>
<td>int</td>
<td>navmesh_create ( NavigationMesh mesh, Transform xform, Object owner=NULL )</td>
</tr>
<tr>
<td>void</td>
<td>navmesh_remove ( int id )</td>
</tr>
<tr>
<td>void</td>
<td>navmesh_set_transform ( int id, Transform xform )</td>
</tr>
<tr>
<td>void</td>
<td>set_up_vector ( Vector3 up )</td>
</tr>
</tbody>
</table>
12.173.3 Member Function Description

- `Vector3 get_closest_point ( Vector3 to_point )`
- `Vector3 get_closest_point_normal ( Vector3 to_point )`
- `Object get_closest_point_owner ( Vector3 to_point )`
- `Vector3 get_closest_point_to_segment ( Vector3 start, Vector3 end, bool use_collision=false )`
- `Vector3Array get_simple_path ( Vector3 start, Vector3 end, bool optimize=true )`
- `Vector3 get_up_vector ( ) const`
- `int navmesh_create ( NavigationMesh mesh, Transform xform, Object owner=NULL )`
- `void navmesh_remove ( int id )`
- `void navmesh_set_transform ( int id, Transform xform )`
- `void set_up_vector ( Vector3 up )`

12.174 Navigation2D

Inherits: `Node2D < CanvasItem < Node < Object`

Category: Core

12.174.1 Brief Description

12.174.2 Member Functions

<table>
<thead>
<tr>
<th><code>Vector2</code></th>
<th><code>get_closest_point ( Vector2 to_point )</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Object</code></td>
<td><code>get_closest_point_owner ( Vector2 to_point )</code></td>
</tr>
<tr>
<td><code>Vector2Array</code></td>
<td><code>get_simple_path ( Vector2 start, Vector2 end, bool optimize=true )</code></td>
</tr>
<tr>
<td><code>int</code></td>
<td><code>navpoly_create ( NavigationPolygon mesh, Matrix32 xform, Object owner=NULL )</code></td>
</tr>
<tr>
<td><code>void</code></td>
<td><code>navpoly_remove ( int id )</code></td>
</tr>
<tr>
<td><code>void</code></td>
<td><code>navpoly_set_transform ( int id, Matrix32 xform )</code></td>
</tr>
</tbody>
</table>

12.174.3 Member Function Description

- `Vector2 get_closest_point ( Vector2 to_point )`
- `Object get_closest_point_owner ( Vector2 to_point )`
- `Vector2Array get_simple_path ( Vector2 start, Vector2 end, bool optimize=true )`
- `int navpoly_create ( NavigationPolygon mesh, Matrix32 xform, Object owner=NULL )`
- `void navpoly_remove ( int id )`
- `void navpoly_set_transform ( int id, Matrix32 xform )`
12.175 NavigationMesh

Inherits: Resource < Reference < Object
Category: Core

12.175.1 Brief Description

12.175.2 Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void add_polygon(IntArray polygon)</td>
<td></td>
</tr>
<tr>
<td>void clear_polygons()</td>
<td></td>
</tr>
<tr>
<td>Void create_from_mesh(Object mesh)</td>
<td></td>
</tr>
<tr>
<td>IntArray get_polygon(int idx)</td>
<td></td>
</tr>
<tr>
<td>int get_polygon_count()</td>
<td>const</td>
</tr>
<tr>
<td>Vector3Array get_vertices()</td>
<td>const</td>
</tr>
<tr>
<td>void set_vertices(Vector3Array vertices)</td>
<td></td>
</tr>
</tbody>
</table>

12.175.3 Member Function Description

• void add_polygon(IntArray polygon)
• void clear_polygons()
• void create_from_mesh(Object mesh)
• IntArray get_polygon(int idx)
• int get_polygon_count() const
• Vector3Array get_vertices() const
• void set_vertices(Vector3Array vertices)

12.176 NavigationMeshInstance

Inherits: Spatial < Node < Object
Category: Core

12.176.1 Brief Description

12.176.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>get_navigation_mesh() const</td>
</tr>
<tr>
<td>bool</td>
<td>is_enabled() const</td>
</tr>
<tr>
<td>void</td>
<td>set_enabled(bool enabled)</td>
</tr>
<tr>
<td>void</td>
<td>set_navigation_mesh(Object navmesh)</td>
</tr>
</tbody>
</table>
12.176.3 Member Function Description

- `Object get_navigation_mesh() const`
- `bool is_enabled() const`
- `void set_enabled(bool enabled)`
- `void set_navigation_mesh(Object navmesh)`

12.177 NavigationPolygon

Inherits: `Resource < Reference < Object`

Category: Core

12.177.1 Brief Description

12.177.2 Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void add_outline(Vector2Array outline)</code></td>
<td></td>
</tr>
<tr>
<td><code>void add_outline_at_index(Vector2Array outline, int index)</code></td>
<td></td>
</tr>
<tr>
<td><code>void add_polygon(IntArray polygon)</code></td>
<td></td>
</tr>
<tr>
<td><code>void clear_outlines()</code></td>
<td></td>
</tr>
<tr>
<td><code>Vector2Array get_outline(int idx) const</code></td>
<td></td>
</tr>
<tr>
<td><code>int get_outline_count() const</code></td>
<td></td>
</tr>
<tr>
<td><code>IntArray get_polygon(int idx)</code></td>
<td></td>
</tr>
<tr>
<td><code>int get_polygon_count() const</code></td>
<td></td>
</tr>
<tr>
<td><code>Vector2Array get_vertices() const</code></td>
<td></td>
</tr>
<tr>
<td><code>void make_polygons_from_outlines()</code></td>
<td></td>
</tr>
<tr>
<td><code>void remove_outline(int idx)</code></td>
<td></td>
</tr>
<tr>
<td><code>void set_outline(int idx, Vector2Array outline)</code></td>
<td></td>
</tr>
<tr>
<td><code>void set_vertices(Vector2Array vertices)</code></td>
<td></td>
</tr>
</tbody>
</table>

12.177.3 Member Function Description

- `void add_outline(Vector2Array outline)`
- `void add_outline_at_index(Vector2Array outline, int index)`
- `void add_polygon(IntArray polygon)`
- `void clear_outlines()`
- `void clear_polygons()`
- `Vector2Array get_outline(int idx) const`
- `int getoutline_count() const`
- `IntArray get_polygon(int idx)`
- `int get_polygon_count() const`
- `Vector2Array get_vertices() const`
• void `make_polygons_from_outlines` ( )
• void `remove_outline` ( `int` idx )
• void `set_outline` ( `int` idx, `Vector2Array` outline )
• void `set_vertices` ( `Vector2Array` vertices )

12.178 NavigationPolygonInstance

Inherits: `Node2D < CanvasItem < Node < Object`

Category: Core

12.178.1 Brief Description

12.178.2 Member Functions

<table>
<thead>
<tr>
<th><code>NavigationPolygon</code></th>
<th><code>get_navigation_polygon</code> ( ) const</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>bool</code></td>
<td><code>is_enabled</code> ( ) const</td>
</tr>
<tr>
<td><code>void</code></td>
<td><code>set_enabled</code> ( <code>bool</code> enabled )</td>
</tr>
<tr>
<td><code>void</code></td>
<td><code>set_navigation_polygon</code> ( <code>NavigationPolygon</code> navpoly )</td>
</tr>
</tbody>
</table>

12.178.3 Member Function Description

- `NavigationPolygon get_navigation_polygon` ( ) const
- `bool is_enabled` ( ) const
- `void set_enabled` ( `bool` enabled )
- `void set_navigation_polygon` ( `NavigationPolygon` navpoly )

12.179 Nil

Category: Built-In Types
12.179.1 Brief Description

12.179.2 Member Functions

| void | Nil ( ColorArray from ) |
| void | Nil ( Vector3Array from ) |
| void | Nil ( Vector2Array from ) |
| void | Nil ( StringArray from ) |
| void | Nil ( RealArray from ) |
| void | Nil ( IntArray from ) |
| void | Nil ( RawArray from ) |
| void | Nil ( Array from ) |
| void | Nil ( Dictionary from ) |
| void | Nil ( InputEvent from ) |
| void | Nil ( Object from ) |
| void | Nil ( RID from ) |
| void | Nil ( NodePath from ) |
| void | Nil ( Image from ) |
| void | Nil ( Color from ) |
| void | Nil ( Transform from ) |
| void | Nil ( Matrix3 from ) |
| void | Nil ( AABB from ) |
| void | Nil ( Quat from ) |
| void | Nil ( Plane from ) |
| void | Nil ( Matrix32 from ) |
| void | Nil ( Vector3 from ) |
| void | Nil ( Rect2 from ) |
| void | Nil ( Vector2 from ) |
| void | Nil ( String from ) |
| void | Nil ( float from ) |
| void | Nil ( int from ) |
| void | Nil ( bool from ) |

12.179.3 Member Function Description

- void Nil ( ColorArray from )
- void Nil ( Vector3Array from )
- void Nil ( Vector2Array from )
- void Nil ( StringArray from )
- void Nil ( RealArray from )
- void Nil ( IntArray from )
- void Nil ( RawArray from )
- void Nil ( Array from )
- void Nil ( Dictionary from )
- void Nil ( InputEvent from )
- void Nil ( Object from )
Godot Engine Documentation, Release 2.1

• void Nil ( RID from )
• void Nil ( NodePath from )
• void Nil ( Image from )
• void Nil ( Color from )
• void Nil ( Transform from )
• void Nil ( Matrix3 from )
• void Nil ( AABB from )
• void Nil ( Quat from )
• void Nil ( Plane from )
• void Nil ( Matrix32 from )
• void Nil ( Vector3 from )
• void Nil ( Rect2 from )
• void Nil ( Vector2 from )
• void Nil ( String from )
• void Nil ( float from )
• void Nil ( int from )
• void Nil ( bool from )

12.180 Node

Inherits: Object

Inherited By: Viewport, Timer, CanvasLayer, EventPlayer, SoundRoomParams, Tween, Spatial, AnimationPlayer, EditorPlugin, ResourcePreloader, AnimationTreePlayer, SamplePlayer, InstancePlaceholder, HTTPRequest, StreamPlayer, CanvasItem

Category: Core

12.180.1 Brief Description

Base class for all the scene elements.

12.180.2 Member Functions

<table>
<thead>
<tr>
<th>void</th>
<th>_enter_tree () virtual</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>_exit_tree () virtual</td>
</tr>
<tr>
<td>void</td>
<td>_fixed_process ( float delta ) virtual</td>
</tr>
<tr>
<td>void</td>
<td>_input ( InputEvent event ) virtual</td>
</tr>
<tr>
<td>void</td>
<td>_process ( float delta ) virtual</td>
</tr>
<tr>
<td>void</td>
<td>_ready () virtual</td>
</tr>
<tr>
<td>void</td>
<td>_unhandled_input ( InputEvent event ) virtual</td>
</tr>
</tbody>
</table>

Continued on next page
Table 12.20 – continued from previous page

<table>
<thead>
<tr>
<th>Function</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void _unhandled_key_input (InputEvent key_event)</code></td>
<td>virtual</td>
</tr>
<tr>
<td><code>void add_child (Node node, bool legible_unique_name=false)</code></td>
<td></td>
</tr>
<tr>
<td><code>void add_to_group (String group, bool persistent=false)</code></td>
<td></td>
</tr>
<tr>
<td><code>bool can_process ()</code></td>
<td>const</td>
</tr>
<tr>
<td><code>Node duplicate (bool use_instancing=false, int flags=7)</code></td>
<td>const</td>
</tr>
<tr>
<td><code>Node find_node (String mask, bool recursive=true, bool owned=true)</code></td>
<td>const</td>
</tr>
<tr>
<td><code>Node get_child (int idx)</code></td>
<td>const</td>
</tr>
<tr>
<td><code>int get_child_count ()</code></td>
<td>const</td>
</tr>
<tr>
<td><code>Array get_children ()</code></td>
<td>const</td>
</tr>
<tr>
<td><code>String get_filename ()</code></td>
<td>const</td>
</tr>
<tr>
<td><code>float get_fixed_process_delta_time ()</code></td>
<td>const</td>
</tr>
<tr>
<td><code>Array get_groups ()</code></td>
<td>const</td>
</tr>
<tr>
<td><code>int get_index ()</code></td>
<td>const</td>
</tr>
<tr>
<td><code>String get_name ()</code></td>
<td>const</td>
</tr>
<tr>
<td><code>Node get_node (NodePath path)</code></td>
<td>const</td>
</tr>
<tr>
<td><code>Array get_node_and_resource (NodePath path)</code></td>
<td>const</td>
</tr>
<tr>
<td><code>Node get_owner ()</code></td>
<td>const</td>
</tr>
<tr>
<td><code>Node get_parent ()</code></td>
<td>const</td>
</tr>
<tr>
<td><code>NodePath get_path ()</code></td>
<td>const</td>
</tr>
<tr>
<td><code>NodePath get_path_to (Node node)</code></td>
<td>const</td>
</tr>
<tr>
<td><code>int get_pause_mode ()</code></td>
<td>const</td>
</tr>
<tr>
<td><code>int get_position_in_parent ()</code></td>
<td>const</td>
</tr>
<tr>
<td><code>float get_process_delta_time ()</code></td>
<td>const</td>
</tr>
<tr>
<td><code>bool get_scene_instance_load_placeholder ()</code></td>
<td>const</td>
</tr>
<tr>
<td><code>SceneTree get_tree ()</code></td>
<td>const</td>
</tr>
<tr>
<td><code>Object get_viewport ()</code></td>
<td>const</td>
</tr>
<tr>
<td><code>bool has_node (NodePath path)</code></td>
<td>const</td>
</tr>
<tr>
<td><code>bool has_node_and_resource (NodePath path)</code></td>
<td>const</td>
</tr>
<tr>
<td><code>bool is_a_parent_of (Node node)</code></td>
<td>const</td>
</tr>
<tr>
<td><code>bool is_displayed_folded ()</code></td>
<td>const</td>
</tr>
<tr>
<td><code>bool is_fixed_processing ()</code></td>
<td>const</td>
</tr>
<tr>
<td><code>bool is_greater_than (Node node)</code></td>
<td>const</td>
</tr>
<tr>
<td><code>bool is_in_group (String group)</code></td>
<td>const</td>
</tr>
<tr>
<td><code>bool is_inside_tree ()</code></td>
<td>const</td>
</tr>
<tr>
<td><code>bool is_processing ()</code></td>
<td>const</td>
</tr>
<tr>
<td><code>bool is_processing_input ()</code></td>
<td>const</td>
</tr>
<tr>
<td><code>bool is_processing_unhandled_input ()</code></td>
<td>const</td>
</tr>
<tr>
<td><code>bool is_processing_unhandled_key_input ()</code></td>
<td>const</td>
</tr>
<tr>
<td><code>void move_child (Node child_node, int to_pos)</code></td>
<td></td>
</tr>
<tr>
<td><code>void print_stray_nodes ()</code></td>
<td></td>
</tr>
<tr>
<td><code>void print_tree ()</code></td>
<td></td>
</tr>
<tr>
<td><code>void propagate_notification (int what)</code></td>
<td></td>
</tr>
<tr>
<td><code>void queue_free ()</code></td>
<td></td>
</tr>
<tr>
<td><code>void raise ()</code></td>
<td></td>
</tr>
<tr>
<td><code>void remove_and_skip ()</code></td>
<td></td>
</tr>
<tr>
<td><code>void remove_child (Node node)</code></td>
<td></td>
</tr>
<tr>
<td><code>void remove_from_group (String group)</code></td>
<td></td>
</tr>
<tr>
<td><code>void replace_by (Node node, bool keep_data=false)</code></td>
<td></td>
</tr>
<tr>
<td><code>void set_display_folded (bool fold)</code></td>
<td></td>
</tr>
<tr>
<td><code>void set_filename (String filename)</code></td>
<td></td>
</tr>
</tbody>
</table>

Continued on next page
void set_fixed_process(bool enable)
void set_name(String name)
void set_owner(Node owner)
void set_pause_mode(int mode)
void set_process(bool enable)
void set_process_input(bool enable)
void set_process_unhandled_input(bool enable)
void set_process_unhandled_key_input(bool enable)
void set_scene_instance_load_placeholder(bool load_placeholder)

### 12.180.3 Signals

- enter_tree()
- exit_tree()
- renamed()

Emitted when the node is renamed.

### 12.180.4 Numeric Constants

- NOTIFICATION_ENTER_TREE = 10
- NOTIFICATION_EXIT_TREE = 11
- NOTIFICATION_MOVED_IN_PARENT = 12
- NOTIFICATION_READY = 13
- NOTIFICATION_FIXED_PROCESS = 16
- NOTIFICATION_PROCESS = 17 — Notification received every frame when the process flag is set (see set_process).
- NOTIFICATION_PARENTED = 18 — Notification received when a node is set as a child of another node. Note that this doesn’t mean that a node entered the Scene Tree.
- NOTIFICATION_UNPARENTED = 19 — Notification received when a node is unparented (parent removed it from the list of children).
- NOTIFICATION_PAUSED = 14
- NOTIFICATION_UNPAUSED = 15
- NOTIFICATION_INSTANCED = 20
- NOTIFICATION_DRAG_BEGIN = 21
- NOTIFICATION_DRAG_END = 22
- PAUSE_MODE_INHERIT = 0
- PAUSE_MODE_STOP = 1
- PAUSE_MODE_PROCESS = 2
- DUPLICATE_SIGNALS = 1
- DUPLICATE_GROUPS = 2
- DUPLICATE_SCRIPTS = 4
12.180.5 Description

Nodes are the base bricks with which Godot games are developed. They can be set as children of other nodes, resulting in a tree arrangement. A given node can contain any number of nodes as children (but there is only one scene tree root node) with the requirement that all siblings (direct children of a node) should have unique names.

Any tree of nodes is called a scene. Scenes can be saved to the disk and then instanced into other scenes. This allows for very high flexibility in the architecture and data model of the projects. Nodes can optionally be added to groups. This makes it easy to reach a number of nodes from the code (for example an “enemies” group) to perform grouped actions.

Scene tree: The SceneTree contains the active tree of nodes. When a node is added to the scene tree, it receives the NOTIFICATION_ENTER_TREE notification and its _enter_tree callback is triggered. Children nodes are always added after their parent node, i.e. the _enter_tree callback of a parent node will be triggered before its child’s.

Once all nodes have been added in the scene tree, they receive the NOTIFICATION_READY notification and their respective _ready callbacks are triggered. For groups of nodes, the _ready callback is called in reverse order, from the children up to the parent nodes.

It means that when adding a scene to the scene tree, the following order will be used for the callbacks: _enter_tree of the parent, _enter_tree of the children, _ready of the children and finally _ready of the parent (and that recursively for the whole scene).

Processing: Nodes can be set to the “process” state, so that they receive a callback on each frame requesting them to process (do something). Normal processing (callback _process, toggled with set_process) happens as fast as possible and is dependent on the frame rate, so the processing time delta is variable. Fixed processing (callback _fixed_process, toggled with set_fixed_process) happens a fixed amount of times per second (by default 60) and is useful to link itself to the physics.

Nodes can also process input events. When set, the _input function will be called for each input that the program receives. In many cases, this can be overkill (unless used for simple projects), and the _unhandled_input function might be preferred; it is called when the input event was not handled by anyone else (typically, GUI Control nodes), ensuring that the node only receives the events that were meant for it.

To keep track of the scene hierarchy (especially when instancing scenes into other scenes), an “owner” can be set for the node with set_owner. This keeps track of who instanced what. This is mostly useful when writing editors and tools, though.

Finally, when a node is freed with free or queue_free, it will also free all its children.

12.180.6 Member Function Description

• void _enter_tree ( ) virtual

Called when the node enters the SceneTree (e.g. upon instancing, scene changing or after calling add_child in a script). If the node has children, its _enter_tree callback will be called first, and then that of the children.

Corresponds to the NOTIFICATION_ENTER_TREE notification in Object.notification.

• void _exit_tree ( ) virtual

Called when the node leaves the SceneTree (e.g. upon freeing, scene changing or after calling remove_child in a script). If the node has children, its _exit_tree callback will be called last, after all its children have left the tree.

Corresponds to the NOTIFICATION_EXIT_TREE notification in Object.notification.

• void _fixed_process ( float delta ) virtual

Called during the fixed processing step of the main loop. Fixed processing means that the frame rate is synced to the physics, i.e. the delta variable should be constant.
It is only called if fixed processing has been enabled with `set_fixed_process`.
Corresponds to the NOTIFICATION_FIXED_PROCESS notification in `Object._notification`.

- **void **_input ( InputEvent event ) virtual

Called for every input event.
It has to be enabled with `set_process_input` or the corresponding property in the inspector.

- **void **_process ( float delta ) virtual

Called during the processing step of the main loop. Processing happens at every frame and as fast as possible, so the `delta` time since the previous frame is not constant.
It is only called if processing has been enabled with `set_process`.
Corresponds to the NOTIFICATION_PROCESS notification in `Object._notification`.

- **void **_ready ( ) virtual

Called when the node is “ready”, i.e. when both the node and its children have entered the scene tree. If the node has children, their _ready callback gets triggered first, and the node will receive the ready notification only afterwards.
Corresponds to the NOTIFICATION_READY notification in `Object._notification`.

- **void **_unhandled_input ( InputEvent event ) virtual

Called for every input event that has not already been handled by another node.
It has to be enabled with `set_process_unhandled_input` or the corresponding property in the inspector.

- **void **_unhandled_key_input ( InputEvent key_event ) virtual

Called for every key input event that has not already been handled by another node.
It has to be enabled with `set_process_unhandled_key_input` or the corresponding property in the inspector.

- **void **add_child ( Node node, bool legible_unique_name=false )

Add a child Node. Nodes can have as many children as they want, but every child must have a unique name. Children nodes are automatically deleted when the parent node is deleted, so deleting a whole scene is performed by deleting its topmost node.

The optional boolean argument enforces creating child nodes with human-readable names, based on the name of the node being instanced instead of its type only.

- **bool **can_process ( ) const

Return true if the node can process, i.e. whether its pause mode allows processing while the scene tree is paused (see `set_pause_mode`). Always returns true if the scene tree is not paused, and false if the node is not in the tree. FIXME: Why FAIL_COND?

- **Node **duplicate ( bool use_instancing=false, int flags=7 ) const

Duplicate the node, returning a new Node. If use_instancing is true, the duplicated node will be a new instance of the original PackedScene, if not it will be an independent node.

The flags are used to define what attributes (groups, signals, scripts) should be duplicated, as specified by the DUPLICATE_* constants. The default value is all of them.

- **Node **find_node ( String mask, bool recursive=true, bool owned=true ) const
Find a descendant of this node whose name matches `mask` as in `String.match` (i.e. case sensitive, but ‘*’ matches zero or more characters and ‘?’ matches any single character except ‘.’). Note that it does not match against the full path, just against individual node names.

- **Node get_child ( int idx ) const**
  Return a child node by its index (see `get_child_count`). This method is often used for iterating all children of a node.

- **int get_child_count ( ) const**
  Return the amount of child nodes.

- **Array get_children ( ) const**
  Return an array of references (`Node`) to the child nodes.

- **String get_filename ( ) const**
  Return a filename that may be contained by the node. When a scene is instanced from a file, it topmost node contains the filename from where it was loaded (see `set_filename`).

- **float get_fixed_process_delta_time ( ) const**
  Return the time elapsed since the last fixed frame (see `_fixed_process`). This is always the same in fixed processing unless the frames per second is changed in `OS`.

- **Array get_groups ( ) const**
  Return an array listing the groups that the node is part of.

- **int get_index ( ) const**
  Get the node index, i.e. its position among the siblings of its parent.

- **String get_name ( ) const**
  Return the name of the node. This name is unique among the siblings (other child nodes from the same parent).

- **Node get_node ( NodePath path ) const**
  Fetch a node. The `NodePath` must be valid (or else an error will be raised) and can be either the path to child node, a relative path (from the current node to another node), or an absolute path to a node.

  Note: fetching absolute paths only works when the node is inside the scene tree (see `is_inside_tree`).

  **Example:** Assume your current node is Character and the following tree:

  ```
  /root
  /root/Character
  /root/Character/Sword
  /root/Character/Backpack/Dagger
  /root/MyGame
  /root/Swamp/Alligator
  /root/Swamp/Mosquito
  /root/Swamp/Goblin
  ```

  Possible paths are:

  ```
  get_node("Sword")
  get_node("Backpack/Dagger")
  get_node("../Swamp/Alligator")
  get_node("/root/MyGame")
  ```

- **Array get_node_and_resource ( NodePath path )**

- **Node get_owner ( ) const**
Get the node owner (see set_owner).

- **Node** get_parent ( ) const

Return the parent node of the current node, or an empty *Node* if the node lacks a parent.

- **NodePath** get_path ( ) const

Return the absolute path of the current node. This only works if the current node is inside the scene tree (see is_inside_tree).

- **NodePath** get_path_to ( *Node* node ) const

Return the relative path from the current node to the specified node in “node” argument. Both nodes must be in the same scene, or else the function will fail.

- **int** get_pause_mode ( ) const

- **int** get_position_in_parent ( ) const

- **float** get_process_delta_time ( ) const

Return the time elapsed (in seconds) since the last process callback. This is almost always different each time.

- **bool** get_scene_instance_load_placeholder ( ) const

- **SceneTree** get_tree ( ) const

- **Object** get_viewport ( ) const

- **bool** has_node ( *NodePath* path ) const

- **bool** has_node_and_resource ( *NodePath* path ) const

- **bool** is_a_parent_of ( *Node* node ) const

Return true if the “node” argument is a direct or indirect child of the current node, otherwise return false.

- **bool** is_displayed_folded ( ) const

- **bool** is_fixed_processing ( ) const

Return true if fixed processing is enabled (see set_fixed_process).

- **bool** is_greater_than ( *Node* node ) const

Return true if “node” occurs later in the scene hierarchy than the current node, otherwise return false.

- **bool** is_in_group ( *String* group ) const

- **bool** is_inside_tree ( ) const

- **bool** is_processing ( ) const

Return whether processing is enabled in the current node (see set_process).

- **bool** is_processing_input ( ) const

Return true if the node is processing input (see set_process_input).

- **bool** is_processing_unhandled_input ( ) const

Return true if the node is processing unhandled input (see set_process_unhandled_input).

- **bool** is_processing_unhandled_key_input ( ) const

- **void** move_child ( *Node* child_node, int to_pos )

Move a child node to a different position (order) amongst the other children. Since calls, signals, etc are performed by tree order, changing the order of children nodes may be useful.
• void `print_stray_nodes()`

Print the scene to stdout. Used mainly for debugging purposes.

• void `print_tree()`

• void `propagate_notification(int what)`

Notify the current node and all its children recursively by calling notification() in all of them.

• void `queue_free()`

• void `raise()`

Move this node to the top of the array of nodes of the parent node. This is often useful on GUIs (`Control`), because their order of drawing fully depends on their order in the tree.

• void `remove_and_skip()`

Remove a node and set all its children as children of the parent node (if exists). All even subscriptions that pass by the removed node will be unsubscribed.

• void `remove_child(Node node)`

Remove a child `Node`. Node is NOT deleted and will have to be deleted manually.

• void `remove_from_group(String group)`

Remove a node from a group.

• void `replace_by(Node node, bool keep_data=false)`

Replace a node in a scene by a given one. Subscriptions that pass through this node will be lost.

• void `set_display_folded(bool fold)`

• void `set_filename(String filename)`

A node can contain a filename. This filename should not be changed by the user, unless writing editors and tools. When a scene is instanced from a file, it topmost node contains the filename from where it was loaded.

• void `set_fixed_process(bool enable)`

Enables or disables node fixed framerate processing. When a node is being processed, it will receive a NOTIFICATION_PROCESS at a fixed (usually 60 fps, check `OS` to change that) interval (and the `_fixed_process` callback will be called if exists). It is common to check how much time was elapsed since the previous frame by calling `get_fixed_process_delta_time`.

• void `set_name(String name)`

Set the name of the `Node`. Name must be unique within parent, and setting an already existing name will cause for the node to be automatically renamed.

• void `set_owner(Node owner)`

Set the node owner. A node can have any other node as owner (as long as a valid parent, grandparent, etc ascending in the tree). When saving a node (using SceneSaver) all the nodes it owns will be saved with it. This allows to create complex SceneTrees, with instancing and subinstancing.

• void `set_pause_mode(int mode)`

• void `set_process(bool enable)`

Enables or disables node processing. When a node is being processed, it will receive a NOTIFICATION_PROCESS on every drawn frame (and the `_process` callback will be called if exists). It is common to check how much time was elapsed since the previous frame by calling `get_process_delta_time`.

• void `set_process_input(bool enable)`
Enable input processing for node. This is not required for GUI controls! It hooks up the node to receive all input (see \_input).

- void set_process_unhandled_input ( bool enable )

Enable unhandled input processing for node. This is not required for GUI controls! It hooks up the node to receive all input that was not previously handled before (usually by a Control). (see \_unhandled_input).

- void set_process_unhandled_key_input ( bool enable )

- void set_scene_instance_load_placeholder ( bool load_placeholder )

### 12.181 Node2D

**Inherits:** CanvasItem < Node < Object

**Inherited By:** RemoteTransform2D, Joint2D, ParticleAttractor2D, CollisionObject2D, VisibilityNotifier2D, TileMap, Navigation2D, CollisionPolygon2D, TouchScreenButton, Particles2D, AnimatedSprite, Light2D, SoundPlayer2D, ViewportSprite, Path2D, Sprite, RayCast2D, CollisionShape2D, NavigationPolygonInstance, PathFollow2D, ParallaxLayer, Polygon2D, Position2D, LightOccluder2D, BackBufferCopy, CanvasModulate, YSort, Camera2D

**Category:** Core

#### 12.181.1 Brief Description

Base node for 2D system.

#### 12.181.2 Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>edit_set_pivot ( Vector2 pivot )</td>
</tr>
<tr>
<td>float</td>
<td>get_angle_to ( Vector2 point ) const</td>
</tr>
<tr>
<td>Vector2</td>
<td>get_global_pos () const</td>
</tr>
<tr>
<td>float</td>
<td>get_global_rot () const</td>
</tr>
<tr>
<td>float</td>
<td>get_global_rotd () const</td>
</tr>
<tr>
<td>Vector2</td>
<td>get_global_scale () const</td>
</tr>
<tr>
<td>Vector2</td>
<td>get_pos () const</td>
</tr>
<tr>
<td>Matrix32</td>
<td>get_relative_transform_to_parent ( Object parent ) const</td>
</tr>
<tr>
<td>float</td>
<td>get_rot () const</td>
</tr>
<tr>
<td>float</td>
<td>get_rotd () const</td>
</tr>
<tr>
<td>Vector2</td>
<td>get_scale () const</td>
</tr>
<tr>
<td>int</td>
<td>get_z () const</td>
</tr>
<tr>
<td>void</td>
<td>global_translate ( Vector2 offset )</td>
</tr>
<tr>
<td>bool</td>
<td>is_z_relative () const</td>
</tr>
<tr>
<td>void</td>
<td>look_at ( Vector2 point )</td>
</tr>
<tr>
<td>void</td>
<td>move_local_x ( float delta, bool scaled=false )</td>
</tr>
<tr>
<td>void</td>
<td>move_local_y ( float delta, bool scaled=false )</td>
</tr>
<tr>
<td>void</td>
<td>rotate ( float radians )</td>
</tr>
<tr>
<td>void</td>
<td>scale ( Vector2 ratio )</td>
</tr>
<tr>
<td>void</td>
<td>set_global_pos ( Vector2 pos )</td>
</tr>
<tr>
<td>void</td>
<td>set_global_rot ( float radians )</td>
</tr>
<tr>
<td>void</td>
<td>set_global_rotd ( float degrees )</td>
</tr>
</tbody>
</table>

Continued on next page
12.181.3 Description

Base node for 2D system. Node2D contains a position, rotation and scale, which is used to position and animate. It can alternatively be used with a custom 2D transform (Matrix32). A tree of Node2Ds allows complex hierarchies for animation and positioning.

12.181.4 Member Function Description

- void edit_set_pivot (Vector2 pivot)
  Set the pivot position of the 2D node to ‘pivot’ value. This method is implemented only in some nodes that inherit Node2D.

- float get_angle_to (Vector2 point) const
  Return the rotation angle in radians needed for the 2d node to point at ‘point’ position.

- Vector2 get_global_pos () const
  Return the global position of the 2D node.

- float get_global_rot () const
  Return the global rotation in radians of the 2D node.

- float get_global_rotd () const
  Return the global rotation in degrees of the 2D node.

- Vector2 get_global_scale () const
  Return the global scale of the 2D node.

- Vector2 get_pos () const
  Return the position of the 2D node.

- Matrix32 get_relative_transform_to_parent (Object parent) const
  Return the transform Matrix32 calculated relatively to the parent of this 2D node.

- float get_rot () const
  Return the rotation of the 2D node.

- float get_rotd () const
  Return the rotation in degrees of the 2D node.

- Vector2 get_scale () const
  Return the scale of the 2D node.
• `int get_z ( ) const`
Return the Z-index of the 2D node.

• `void global_translate ( Vector2 offset )`
Apply a global translation of ‘offset’ to the 2D node, starting from its current global position.

• `bool is_z_relative ( ) const`
Return true if the Z-index value of this 2D node is relative to its parent’s. Else, return false.

• `void look_at ( Vector2 point )`
Rotate the 2d node so it points at ‘point’ position.

• `void move_local_x ( float delta, bool scaled=false )`
Apply a local translation on X axis to the 2D node according to the ‘delta’ of the process. If ‘scaled’ is false, the movement is normalized.

• `void move_local_y ( float delta, bool scaled=false )`
Apply a local translation on Y axis to the 2D node according to the ‘delta’ of the process. If ‘scaled’ is false, the movement is normalized.

• `void rotate ( float radians )`
Apply a ‘radians’ rotation to the 2D node, starting from its current rotation.

• `void scale ( Vector2 ratio )`
Apply the ‘ratio’ scale to the 2D node, according to its current scale value.

• `void set_global_pos ( Vector2 pos )`
Set the global position of the 2D node to ‘pos’.

• `void set_global_rot ( float radians )`
Set the global rotation in radians of the 2D node.

• `void set_global_rotd ( float degrees )`
Set the global rotation in degrees of the 2D node.

• `void set_global_scale ( Vector2 scale )`
Set the global scale of the 2D node.

• `void set_global_transform ( Matrix32 xform )`
Set the global transform `Matrix32` of the 2D node.

• `void set_pos ( Vector2 pos )`
Set the position of the 2D node.

• `void set_rot ( float radians )`
Set the rotation of the 2D node.

• `void set_rotd ( float degrees )`
Set the rotation of the 2D node.

• `void set_scale ( Vector2 scale )`
Set the scale of the 2D node.

• `void set_transform ( Matrix32 xform )`
Set the local transform $\text{Matrix32}$ of the 2D node.

- void set_z (int z)

Set the Z-index value of the 2D node.

- void set_z_as_relative (bool enable)

Set the Z-index value as relative to the parent node of this 2D node. Thus, if this 2D node’s Z-index value is 2 and its parent’s effective Z-index is 3, then the effective Z-index value of this 2D node would be $3 + 2 = 5$.

- void translate (Vector2 offset)

Apply a local translation of ‘offset’ to the 2D node, starting from its current local position.

### 12.182 NodePath

**Category:** Built-In Types

#### 12.182.1 Brief Description

Pre-parsed scene tree path.

#### 12.182.2 Member Functions

<table>
<thead>
<tr>
<th>NodePath</th>
<th>NodePath (String from)</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>get_name (int idx)</td>
</tr>
<tr>
<td>int</td>
<td>get_name_count()</td>
</tr>
<tr>
<td>String</td>
<td>get_property()</td>
</tr>
<tr>
<td>String</td>
<td>get_subname (int idx)</td>
</tr>
<tr>
<td>int</td>
<td>get_subname_count()</td>
</tr>
<tr>
<td>bool</td>
<td>is_absolute()</td>
</tr>
<tr>
<td>bool</td>
<td>is_empty()</td>
</tr>
</tbody>
</table>

#### 12.182.3 Description

A pre-parsed relative or absolute path in a scene tree, for use with `Node.get_node` and similar functions. It can reference a node, a resource within a node, or a property of a node or resource. For instance, "Path2D/PathFollow2D/Sprite:texture:size" would refer to the size property of the texture resource on the node named "Sprite" which is a child of the other named nodes in the path. Note that if you want to get a resource, you must end the path with a colon, otherwise the last element will be used as a property name.

You will usually just pass a string to `Node.get_node` and it will be automatically converted, but you may occasionally want to parse a path ahead of time with `NodePath` or the literal syntax @$\text{path}@$. Exporting a `NodePath` variable will give you a node selection widget in the properties panel of the editor, which can often be useful.

A `NodePath` is made up of a list of node names, a list of “subnode” (resource) names, and the name of a property in the final node or resource.
12.182.4 Member Function Description

- **NodePath NodePath ( String from )**
  Create a NodePath from a string, e.g. “Path2D/PathFollow2D/Sprite:texture:size”. A path is absolute if it starts with a slash. Absolute paths are only valid in the global scene tree, not within individual scenes. In a relative path, "." and ". ." indicate the current node and its parent.

- **String get_name ( int idx )**
  Get the node name indicated by `idx` (0 to `get_name_count`)

- **int get_name_count ( )**
  Get the number of node names which make up the path.

- **String get_property ( )**
  Get the path’s property name, or an empty string if the path doesn’t have a property.

- **String get_subname ( int idx )**
  Get the resource name indicated by `idx` (0 to `get_subname_count`)

- **int get_subname_count ( )**
  Get the number of resource names in the path.

- **bool is_absolute ( )**
  Return true if the node path is absolute (not relative).

- **bool is_empty ( )**
  Return true if the node path is empty.

12.183 Object


**Category:** Core

12.183.1 Brief Description

Base class for all non built-in types.

12.183.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>XL_MESSAGE ( String message ) const</td>
</tr>
<tr>
<td>void</td>
<td>_get ( String property ) virtual</td>
</tr>
<tr>
<td>Array</td>
<td>_get_property_list ( ) virtual</td>
</tr>
<tr>
<td>void</td>
<td>_init ( ) virtual</td>
</tr>
<tr>
<td>void</td>
<td>_notification ( int what ) virtual</td>
</tr>
</tbody>
</table>
Table 12.22 – continued from previous page

<table>
<thead>
<tr>
<th>Function/Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void _set(String property, Variant value)</code> virtual</td>
<td>Sets the value of a property</td>
</tr>
<tr>
<td><code>void add_user_signal(String signal, Array arguments=Array())</code></td>
<td>Adds a user-defined signal to the event system</td>
</tr>
<tr>
<td><code>void call(String method, Variant arg0=NULL, Variant arg1=NULL, ...)</code></td>
<td>Calls a method with specified arguments and arguments</td>
</tr>
<tr>
<td><code>void call_deferred(String method, Variant arg0=NULL, Variant arg1=NULL, ...)</code></td>
<td>Calls a method in deferred mode</td>
</tr>
<tr>
<td><code>Variant callv(String method, Array arg_array)</code></td>
<td>Calls a method with a variable number of arguments</td>
</tr>
<tr>
<td><code>bool can_translate_messages()</code></td>
<td>Determines if messages can be translated to the native language</td>
</tr>
<tr>
<td><code>int connect(String signal, Object target, String method, Array binds=Array(), int flags=0)</code></td>
<td>Connects a signal to a method on an object</td>
</tr>
<tr>
<td><code>void disconnect(String signal, Object target, String method)</code></td>
<td>Disconnects a signal from a method on an object</td>
</tr>
<tr>
<td><code>void emit_signal(String signal, Variant arg0=NULL, ...)</code></td>
<td>Emits a signal with specified arguments and arguments</td>
</tr>
<tr>
<td><code>void free()</code></td>
<td>Frees resources used by the object</td>
</tr>
<tr>
<td><code>void get(String property)</code> const</td>
<td>Gets the value of a property</td>
</tr>
<tr>
<td><code>int get_instance_ID()</code> const</td>
<td>Gets the instance ID of the object</td>
</tr>
<tr>
<td><code>Variant get_meta(String name)</code> const</td>
<td>Gets the value of a meta property</td>
</tr>
<tr>
<td><code>StringArray get_meta_list()</code> const</td>
<td>Gets a list of meta properties</td>
</tr>
<tr>
<td><code>Array get_method_list()</code> const</td>
<td>Gets a list of methods available</td>
</tr>
<tr>
<td><code>Array get_property_list()</code> const</td>
<td>Gets a list of properties available</td>
</tr>
<tr>
<td><code>Script get_script()</code> const</td>
<td>Gets the script associated with the object</td>
</tr>
<tr>
<td><code>Array get_signal_connection_list(String signal)</code> const</td>
<td>Gets a list of signal connections for a specified signal</td>
</tr>
<tr>
<td><code>Array get_signal_list()</code> const</td>
<td>Gets a list of signals available</td>
</tr>
<tr>
<td><code>String get_type()</code> const</td>
<td>Gets the type of the object</td>
</tr>
<tr>
<td><code>bool has_meta(String name)</code> const</td>
<td>Determines if a meta property is defined</td>
</tr>
<tr>
<td><code>bool has_method(String method)</code> const</td>
<td>Determines if a method is defined</td>
</tr>
<tr>
<td><code>bool has_user_signal(String signal)</code> const</td>
<td>Determines if a user-defined signal is defined</td>
</tr>
<tr>
<td><code>bool is_blocking_signals()</code> const</td>
<td>Determines if blocking signals are enabled</td>
</tr>
<tr>
<td><code>bool is_connected(String signal, Object target, String method)</code></td>
<td>Determines if a signal is connected on an object</td>
</tr>
<tr>
<td><code>bool is_queued_for_deletion()</code> const</td>
<td>Determines if an object is queued for deletion</td>
</tr>
<tr>
<td><code>bool is_type(String type)</code> const</td>
<td>Determines if the object is of a specified type</td>
</tr>
<tr>
<td><code>void notification(int what, bool reversed=false)</code></td>
<td>Triggers a notification event with specified arguments and reversed mode</td>
</tr>
<tr>
<td><code>void property_list_changed_notify()</code></td>
<td>Notifies that the property list has changed</td>
</tr>
<tr>
<td><code>void set(String property, Variant value)</code></td>
<td>Sets the value of a property</td>
</tr>
<tr>
<td><code>void set_block_signals(bool enable)</code></td>
<td>Enables or disables blocking signals</td>
</tr>
<tr>
<td><code>void set_message_translation(bool enable)</code></td>
<td>Enables or disables message translation</td>
</tr>
<tr>
<td><code>void set_meta(String name, Variant value)</code></td>
<td>Sets the value of a meta property</td>
</tr>
<tr>
<td><code>void set_script(Script script)</code></td>
<td>Sets the script associated with the object</td>
</tr>
<tr>
<td><code>String tr(String message)</code> const</td>
<td>Translates a message to the native language</td>
</tr>
</tbody>
</table>

12.183.3 Signals

- `script_changed()`

12.183.4 Numeric Constants

- `NOTIFICATION_POSTINITIALIZE = 0` — Called right when the object is initialized. Not available in script.
- `NOTIFICATION_PREDELETE = 1` — Called before the object is about to be deleted.
- `CONNECT_DEFERRED = 1` — Connect a signal in deferred mode. This way, signal emissions are stored in a queue, then set on idle time.
- `CONNECT_PERSIST = 2` — Persisting connections are saved when the object is serialized to file.
• **CONNECT_ONESHOT = 4** — One shot connections disconnect themselves after emission.

### 12.183.5 Description

Base class for all non built-in types. Everything not a built-in type starts the inheritance chain from this class.

Objects do not manage memory, if inheriting from one the object will most likely have to be deleted manually (call the `free` function from the script or delete from C++).

Some derivatives add memory management, such as `Reference` (which keeps a reference count and deletes itself automatically when no longer referenced) and `Node`, which deletes the children tree when deleted.

Objects export properties, which are mainly useful for storage and editing, but not really so much in programming. Properties are exported in `_get_property_list` and handled in `_get` and `_set`. However, scripting languages and C++ have simpler means to export them.

Objects also receive notifications (`_notification`). Notifications are a simple way to notify the object about simple events, so they can all be handled together.

### 12.183.6 Member Function Description

- **String XL_MESSAGE ( String message ) const**

  Deprecated, will go away.

- **void _get ( String property ) virtual**

  Return a property, return null if the property does not exist.

- **Array _get_property_list ( ) virtual**

  Return the property list, array of dictionaries, dictionaries must contain: name:String, type:int (see TYPE_* enum in globals) and optionally: hint:int (see PROPERTY_HINT_* in globals), hint_string:String, usage:int (see PROPERTY_USAGE_* in globals).

- **void _init ( ) virtual**

- **void _notification ( int what ) virtual**

  Notification request, the notification id is received.

- **void _set ( String property, Variant value ) virtual**

  Set a property. Return true if the property was found.

- **void add_user_signal ( String signal, Array arguments=Array() )**

  Add a user signal (can be added anytime). Arguments are optional, but can be added as an array of dictionaries, each containing “name” and “type” (from `@Global Scope TYPE_*`).

- **void call ( String method, Variant arg0=NULL, Variant arg1=NULL, Variant arg2=NULL, Variant arg3=NULL, Variant arg4=NULL, Variant arg5=NULL, Variant arg6=NULL, Variant arg7=NULL, Variant arg8=NULL, Variant arg9=NULL )**

  Call a function in the object. Result is returned.

- **void call_deferred ( String method, Variant arg0=NULL, Variant arg1=NULL, Variant arg2=NULL, Variant arg3=NULL, Variant arg4= NULL )**

  Create and store a function in the object. The call will take place on idle time.

- **Variant callv ( String method, Array arg_array )**
• `bool can_translate_messages()` const
Return true if this object can translate strings.

• `int connect()`
  
  `String signal`, `Object target`, `String method`, `Array binds=Array()`, `int flags=0`

Connect a signal to a method at a target (member function). Binds are optional and are passed as extra arguments to the call. Flags specify optional deferred or one shot connections, see enum CONNECT_*. A signal can only be connected once to a method, and it will throw an error if already connected. If you want to avoid this, use `is_connected` to check.

• `void disconnect()`
  
  `String signal`, `Object target`, `String method`
Disconnect a signal from a method.

• `void emit_signal()`
  
  `String signal`, `Variant arg0=NULL`, `Variant arg1=NULL`, `Variant arg2=NULL`, `Variant arg3=NULL`, `Variant arg4=NULL`

Emit a signal. Arguments are passed in an array.

• `void free()`

Get a property from the object.

• `int get_instance_ID()` const
Return the instance ID. All objects have a unique instance ID.

• `Variant get_meta()` const
Return a metadata from the object.

• `StringArray get_meta_list()` const
Return the list of metadata in the object.

• `Array get_method_list()` const

• `Array get_property_list()` const
Return the list of properties as an array of dictionaries, dictionaries contain: name:String, type:int (see TYPE_* enum in globals) and optionally: hint:int (see PROPERTY_HINT_* in globals), hint_string:String, usage:int (see PROPERTY_USAGE_* in globals).

• `Script get_script()` const
Return the object script (or null if it doesn’t have one).

• `Array get_signal_connection_list()`
  
  `String signal`

• `Array get_signal_list()` const
Return the list of signals as an array of dictionaries.

• `String get_type()` const
Return the type of the object as a string.

• `bool has_meta()` const
Return true if a metadata is found with the requested name.

• `bool has_method()` const

• `bool has_user_signal()` const

• `bool is_blocking_signals()` const
Return true if signal emission blocking is enabled.

- `bool is_connected ( String signal, Object target, String method ) const`

Return true if a connection exists for a given signal and target/method.

- `bool is_queued_for_deletion ( ) const`
- `bool is_type ( String type ) const`

Check the type of the object against a string (including inheritance).

- `void notification ( int what, bool reversed=false )`

Notify the object of something.

- `void property_list_changed_notify ( )`
- `void set ( String property, Variant value )`

Set property into the object.

- `void set_block_signals ( bool enable )`

If set to true, signal emission is blocked.

- `void set_message_translation ( bool enable )`

Set true if this object can translate strings (in calls to tr()). Default is true.

- `void set_meta ( String name, Variant value )`

Set a metadata into the object. Metadata is serialized. Metadata can be *anything*.

- `void set_script ( Script script )`

Set a script into the object, scripts extend the object functionality.

- `String tr ( String message ) const`

Translate a message. Only works in message translation is enabled (which is by default). See `set_message_translation`.

### 12.184 OccluderPolygon2D

**Inherits:** Resource < Reference < Object

**Category:** Core

#### 12.184.1 Brief Description

#### 12.184.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td><code>get_cull_mode ( ) const</code></td>
</tr>
<tr>
<td>Vector2Array</td>
<td><code>get_polygon ( ) const</code></td>
</tr>
<tr>
<td>bool</td>
<td><code>is_closed ( ) const</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_closed ( bool closed )</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_cull_mode ( int cull_mode )</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_polygon ( Vector2Array polygon )</code></td>
</tr>
</tbody>
</table>
12.184.3 Numeric Constants

- CULL_DISABLED = 0
- CULL_CLOCKWISE = 1
- CULL_COUNTER_CLOCKWISE = 2

12.184.4 Member Function Description

- int get_cull_mode() const
- Vector2Array get_polygon() const
- bool is_closed() const
- void set_closed(bool closed)
- void set_cull_mode(int cull_mode)
- void set_polygon(Vector2Array polygon)

12.185 OmniLight

Inherits: Light < VisualInstance < Spatial < Node < Object

Category: Core

12.185.1 Brief Description

OmniDirectional Light, such as a light bulb or a candle.

12.185.2 Description

An OmniDirectional light is a type of Light node that emits lights in all directions. The light is attenuated through the distance and this attenuation can be configured by changing the energy, radius and attenuation parameters of Light. TODO: Image of an omnilight.

12.186 OptionButton

Inherits: Button < BaseButton < Control < CanvasItem < Node < Object

Category: Core

12.186.1 Brief Description

Button control that provides selectable options when pressed.
12.186.2 Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>void add_icon_item</td>
<td>Texture texture, String label, int id</td>
</tr>
<tr>
<td>void add_item</td>
<td>String label, int id=-1</td>
</tr>
<tr>
<td>void add_separator</td>
<td></td>
</tr>
<tr>
<td>void clear</td>
<td></td>
</tr>
<tr>
<td>int get_item_ID</td>
<td>int idx const</td>
</tr>
<tr>
<td>int get_item_count</td>
<td>(const)</td>
</tr>
<tr>
<td>Texture get_item_icon</td>
<td>int idx const</td>
</tr>
<tr>
<td>void get_item_metadata</td>
<td>int idx const</td>
</tr>
<tr>
<td>String get_item_text</td>
<td>int idx const</td>
</tr>
<tr>
<td>int get_selected</td>
<td>(const)</td>
</tr>
<tr>
<td>void get_selected_metadata</td>
<td>(const)</td>
</tr>
<tr>
<td>bool is_item_disabled</td>
<td>int idx const</td>
</tr>
<tr>
<td>void remove_item</td>
<td>int idx</td>
</tr>
<tr>
<td>void select</td>
<td>int idx</td>
</tr>
<tr>
<td>void set_item_ID</td>
<td>int idx, int id</td>
</tr>
<tr>
<td>void set_item_disabled</td>
<td>int idx, bool disabled</td>
</tr>
<tr>
<td>void set_item_icon</td>
<td>int idx, Texture texture</td>
</tr>
<tr>
<td>void set_item_metadata</td>
<td>int idx, Variant metadata</td>
</tr>
<tr>
<td>void set_item_text</td>
<td>int idx, String text</td>
</tr>
</tbody>
</table>

12.186.3 Signals

- item_selected (int ID)

This signal is emitted when the current item was changed by the user. ID of the item selected is passed as argument (if no IDs were added, ID will be just the item index).

12.186.4 Description

ToggleButton is a type button that provides a selectable list of items when pressed. The item selected becomes the “current” item and is displayed as the button text.

12.186.5 Member Function Description

- void add_icon_item (Texture texture, String label, int id)

Add an item, with a “texture” icon, text “label” and (optionally) id. If no “id” is passed, “id” becomes the item index. New items are appended at the end.

- void add_item (String label, int id=-1)

Add an item, with text “label” and (optionally) id. If no “id” is passed, “id” becomes the item index. New items are appended at the end.

- void add_separator ()

Add a separator to the list of items. Separators help to group items. Separator also takes up an index and is appended at the end.

- void clear ()
Clear all the items in the `OptionButton`.

- `int get_item_ID (int idx) const`
  Return the ID of the item at index “idx”.

- `int get_item_count () const`
  Return the amount of items in the `OptionButton`.

- `Texture get_item_icon (int idx) const`
  Return the icon of the item at index “idx”.

- `void get_item_metadata (int idx) const`

- `String get_item_text (int idx) const`
  Return the text of the item at index “idx”.

- `int get_selected () const`
  Return the current item index

- `int get_selected_ID () const`

- `void get_selected_metadata () const`

- `bool is_item_disabled (int idx) const`

- `void remove_item (int idx) const`

- `void select (int idx) const`
  Select an item by index and make it the current item.

- `void set_item_ID (int idx, int id) const`
  Set the ID of an item at index “idx”.

- `void set_item_disabled (int idx, bool disabled) const`

- `void set_item_icon (int idx, Texture texture) const`
  Set the icon of an item at index “idx”.

- `void set_item_metadata (int idx, Variant metadata) const`

- `void set_item_text (int idx, String text) const`
  Set the text of an item at index “idx”.

12.187 OS

Inherits: `Object`

Category: Core

12.187.1 Brief Description

Operating System functions.
12.187.2 Member Functions
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void alert(String text, String title=&quot;Alert!&quot;)</td>
<td>Displays a dialog box with the specified text and title.</td>
</tr>
<tr>
<td>bool can_draw()</td>
<td>Checks if the engine supports drawing operations.</td>
</tr>
<tr>
<td>bool can_use_threads()</td>
<td>Determines if the engine can use threads for asynchronous operations.</td>
</tr>
<tr>
<td>void delay_msec(int msec)</td>
<td>Pauses the engine for a specified number of milliseconds.</td>
</tr>
<tr>
<td>void delay_usec(int usec)</td>
<td>Pauses the engine for a specified number of microseconds.</td>
</tr>
<tr>
<td>void dump_memory_to_file(String file)</td>
<td>Exports the engine's memory state to a file.</td>
</tr>
<tr>
<td>void dump_resources_to_file(String file)</td>
<td>Exports the engine's resources to a file.</td>
</tr>
<tr>
<td>int execute(String path, StringArray arguments, bool blocking, Array output=Array())</td>
<td>Runs an executable with specified arguments and returns output directives.</td>
</tr>
<tr>
<td>bool get_borderless_window()</td>
<td>Checks if the engine supports a borderless window.</td>
</tr>
<tr>
<td>String get_clipboard()</td>
<td>Retrieves the contents of the clipboard.</td>
</tr>
<tr>
<td>StringArray get_cmdline_args()</td>
<td>Returns the command line arguments passed to the engine.</td>
</tr>
<tr>
<td>int get_current_screen()</td>
<td>Returns the currently active screen index.</td>
</tr>
<tr>
<td>String get_custom_level()</td>
<td>Returns the custom level currently active.</td>
</tr>
<tr>
<td>String get_data_dir()</td>
<td>Returns the data directory path.</td>
</tr>
<tr>
<td>Dictionary get_date(bool utc=false)</td>
<td>Returns the current date in various formats.</td>
</tr>
<tr>
<td>Dictionary get_datetime(bool utc=false)</td>
<td>Returns the current datetime in various formats.</td>
</tr>
<tr>
<td>Dictionary get_datetime_from_unix_time(int unix_time_val)</td>
<td>Converts a unix time value to a datetime.</td>
</tr>
<tr>
<td>int get_dynamic_memory_usage()</td>
<td>Returns the current dynamic memory usage.</td>
</tr>
<tr>
<td>Dictionary get_engine_version()</td>
<td>Returns information about the engine's version.</td>
</tr>
<tr>
<td>String get_environment(String environment)</td>
<td>Returns the environment variable with the specified name.</td>
</tr>
<tr>
<td>String get_executable_path()</td>
<td>Returns the path to the executable.</td>
</tr>
<tr>
<td>int get_exit_code()</td>
<td>Returns the exit code of the last executed process.</td>
</tr>
<tr>
<td>int get_frames_drawn()</td>
<td>Returns the number of frames drawn.</td>
</tr>
<tr>
<td>float get_frames_per_second()</td>
<td>Returns the frames per second.</td>
</tr>
<tr>
<td>Array get_fullscreen_mode_list(int screen=0)</td>
<td>Returns a list of supported full-screen modes.</td>
</tr>
<tr>
<td>int get_iterations_per_second()</td>
<td>Returns the current iterations per second.</td>
</tr>
<tr>
<td>String get_latin_keyboard_variant()</td>
<td>Returns the current Latin keyboard variant.</td>
</tr>
<tr>
<td>String get_locale()</td>
<td>Returns the current locale.</td>
</tr>
<tr>
<td>Object get_main_loop()</td>
<td>Returns the main loop object.</td>
</tr>
<tr>
<td>String get_model_name()</td>
<td>Returns the name of the current model.</td>
</tr>
<tr>
<td>int get_process_ID()</td>
<td>Returns the process ID.</td>
</tr>
<tr>
<td>int get_processor_count()</td>
<td>Returns the number of processors.</td>
</tr>
<tr>
<td>String get_scancode_string(int code)</td>
<td>Returns the string representation of the specified scancode.</td>
</tr>
<tr>
<td>int get_screen_count()</td>
<td>Returns the number of screens.</td>
</tr>
<tr>
<td>int get_screen_dpi(int screen=0)</td>
<td>Returns the screen's DPI.</td>
</tr>
<tr>
<td>int get_screen_orientation()</td>
<td>Returns the screen's orientation.</td>
</tr>
<tr>
<td>Vector2 get_screen_position(int screen=0)</td>
<td>Returns the screen's position.</td>
</tr>
<tr>
<td>Vector2 get_screen_size(int screen=0)</td>
<td>Returns the screen's size.</td>
</tr>
<tr>
<td>int get_splash_tick_msec()</td>
<td>Returns the splash screen tick in milliseconds.</td>
</tr>
<tr>
<td>int get_static_memory_peak_usage()</td>
<td>Returns the peak static memory usage.</td>
</tr>
<tr>
<td>int get_static_memory_usage()</td>
<td>Returns the current static memory usage.</td>
</tr>
<tr>
<td>String get_system_dir(int dir)</td>
<td>Returns the system directory path for the specified directory.</td>
</tr>
<tr>
<td>int get_system_time_secs()</td>
<td>Returns the system time in seconds.</td>
</tr>
<tr>
<td>float get_target_fps()</td>
<td>Returns the target frames per second.</td>
</tr>
<tr>
<td>int get_ticks_msec()</td>
<td>Returns the current number of ticks.</td>
</tr>
<tr>
<td>Dictionary get_time(bool utc=false)</td>
<td>Returns the current time in various formats.</td>
</tr>
<tr>
<td>float get_time_scale()</td>
<td>Returns the time scale.</td>
</tr>
<tr>
<td>Dictionary get_time_zone_info()</td>
<td>Returns information about the current time zone.</td>
</tr>
</tbody>
</table>
Table 12.23 – continued from previous page

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td><code>get_unique_ID()</code></td>
</tr>
<tr>
<td>int</td>
<td><code>get_unix_time()</code></td>
</tr>
<tr>
<td>int</td>
<td><code>get_unix_time_from_datetime(Dictionary datetime)</code></td>
</tr>
<tr>
<td>Vector2</td>
<td><code>get_video_mode_size(int screen=0)</code></td>
</tr>
<tr>
<td>Vector2</td>
<td><code>get_window_position()</code></td>
</tr>
<tr>
<td>Vector2</td>
<td><code>get_window_size()</code></td>
</tr>
<tr>
<td>bool</td>
<td><code>has_environment(String environment)</code></td>
</tr>
<tr>
<td>bool</td>
<td><code>has_touchscreen_ui_hint()</code></td>
</tr>
<tr>
<td>bool</td>
<td><code>has_virtual_keyboard()</code></td>
</tr>
<tr>
<td>void</td>
<td><code>hide_virtual_keyboard()</code></td>
</tr>
<tr>
<td>bool</td>
<td><code>is_debug_build()</code></td>
</tr>
<tr>
<td>bool</td>
<td><code>is_in_low_processor_usage_mode()</code></td>
</tr>
<tr>
<td>bool</td>
<td><code>is_keep_screen_on()</code></td>
</tr>
<tr>
<td>bool</td>
<td><code>is_ok_left_and_cancel_right()</code></td>
</tr>
<tr>
<td>bool</td>
<td><code>is_scancode_unicode(int code)</code></td>
</tr>
<tr>
<td>bool</td>
<td><code>is_stdout_verbose()</code></td>
</tr>
<tr>
<td>bool</td>
<td><code>is_video_mode_fullscreen(int screen=0)</code></td>
</tr>
<tr>
<td>bool</td>
<td><code>is_video_mode_resizable(int screen=0)</code></td>
</tr>
<tr>
<td>bool</td>
<td><code>is_vsync_enabled()</code></td>
</tr>
<tr>
<td>bool</td>
<td><code>is_window_fullscreen()</code></td>
</tr>
<tr>
<td>bool</td>
<td><code>is_window_maximized()</code></td>
</tr>
<tr>
<td>bool</td>
<td><code>is_window_minimized()</code></td>
</tr>
<tr>
<td>bool</td>
<td><code>is_windowResizable()</code></td>
</tr>
<tr>
<td>int</td>
<td><code>kill(int pid)</code></td>
</tr>
<tr>
<td>bool</td>
<td><code>native_video_is_playing()</code></td>
</tr>
<tr>
<td>void</td>
<td><code>native_video_pause()</code></td>
</tr>
<tr>
<td>int</td>
<td><code>native_video_play(String path, float volume, String audio_track, String subtitle_track)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>native_video_stop()</code></td>
</tr>
<tr>
<td>void</td>
<td><code>native_video_unpause()</code></td>
</tr>
<tr>
<td>void</td>
<td><code>print_all_resources(String tofile=&quot;&quot;)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>print_all_textures_by_size()</code></td>
</tr>
<tr>
<td>void</td>
<td><code>print_resources_by_type(StringArray types)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>print_resources_in_use(bool short=false)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>request_attention()</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_borderless_window(bool borderless)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_clipboard(String clipboard)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_current_screen(int screen)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_exit_code(int code)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_icon(Image icon)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_iterations_per_second(int iterations_per_second)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_keep_screen_on(bool enabled)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_low_processor_usage_mode(bool enable)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_screen_orientation(int orientation)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_target_fps(int target_fps)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_thread_name(String name)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_time_scale(float time_scale)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_use_file_access_save_and_swap(bool enabled)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_use_vsync(bool enable)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_video_mode(Vector2 size, bool fullscreen, bool resizable, int screen=0)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_window_fullscreen(bool enabled)</code></td>
</tr>
</tbody>
</table>

Continued on next page
### 12.187.3 Numeric Constants

- `DAY_SUNDAY = 0`
- `DAY_MONDAY = 1`
- `DAY_TUESDAY = 2`
- `DAY_WEDNESDAY = 3`
- `DAY_THURSDAY = 4`
- `DAY_FRIDAY = 5`
- `DAY_SATURDAY = 6`
- `MONTH_JANUARY = 1`
- `MONTH_FEBRUARY = 2`
- `MONTH_MARCH = 3`
- `MONTH_APRIL = 4`
- `MONTH_MAY = 5`
- `MONTH_JUNE = 6`
- `MONTH_JULY = 7`
- `MONTH_AUGUST = 8`
- `MONTH_SEPTEMBER = 9`
- `MONTH_OCTOBER = 10`
- `MONTH_NOVEMBER = 11`
- `MONTH_DECEMBER = 12`
- `SCREEN_ORIENTATION_LANDSCAPE = 0`
- `SCREEN_ORIENTATION_PORTRAIT = 1`
- `SCREEN_ORIENTATION_REVERSE_LANDSCAPE = 2`
- `SCREEN_ORIENTATION_REVERSE_PORTRAIT = 3`
- `SCREEN_ORIENTATION_SENSOR_LANDSCAPE = 4`
- `SCREEN_ORIENTATION_SENSOR_PORTRAIT = 5`
- `SCREEN_ORIENTATION_SENSOR = 6`
- `SYSTEM_DIR_DESKTOP = 0`
• SYSTEM_DIR_DCIM = 1
• SYSTEM_DIR_DOCUMENTS = 2
• SYSTEM_DIR_DOWNLOADS = 3
• SYSTEM_DIR_MOVIES = 4
• SYSTEM_DIR_MUSIC = 5
• SYSTEM_DIR_PICTURES = 6
• SYSTEM_DIR_RINGTONES = 7

12.187.4 Description

Operating System functions. OS Wraps the most common functionality to communicate with the host Operating System, such as: mouse grabbing, mouse cursors, clipboard, video mode, date and time, timers, environment variables, execution of binaries, command line, etc.

12.187.5 Member Function Description

• void alert ( String text, String title="Alert!" )
  Displays a modal dialog box utilizing the host OS.

• bool can_draw ( ) const
  Return true if the host OS allows drawing.

• bool can_use_threads ( ) const
  Returns if the current host platform is using multiple threads.

• void delay_msec ( int msec ) const
  Delay executing of the current thread by given milliseconds.

• void delay_usec ( int usec ) const
  Delay executing of the current thread by given microseconds.

• void dump_memory_to_file ( String file )
  Dumps the memory allocation ringlist to a file (only works in debug).
  Entry format per line: “Address - Size - Description”

• void dump_resources_to_file ( String file )
  Dumps all used resources to file (only works in debug).
  Entry format per line: “Resource Type : Resource Location”
  At the end of the file is a statistic of all used Resource Types.

• int execute ( String path, StringArray arguments, bool blocking, Array output=Array() )
  Execute the binary file in given path, optionally blocking until it returns. A process ID is returned.

• int find_scancode_from_string ( String string ) const
  Returns the scancode of the given string (e.g. “Escape”)

• bool get_borderless_window ( ) const
Returns true if the current window is borderless.

- **String** `get_clipboard()` const
  Get clipboard from the host OS.

- **StringArray** `get_cmdline_args()`
  Return the CommandLine passed to the engine.

- **int** `get_current_screen()` const
  Returns the current screen index (0 padded).

- **String** `get_custom_level()` const
  Returns the value of the CommandLine argument “-level”.

- **String** `get_data_dir()` const
  Return the absolute directory path of user data path(user://).

- **Dictionary** `get_date(bool utc=false)` const
  Returns current date as a dictionary of keys: year, month, day, weekday, dst (daylight saving time).

- **Dictionary** `get_datetime(bool utc=false)` const
  Returns current datetime as a dictionary of keys: year, month, day, weekday, dst (daylight saving time), hour, minute, second.

- **Dictionary** `get_datetime_from_unix_time(int unix_time_val)` const
  Get a dictionary of time values when given epoch time.

Dictionary Time values will be a union of values from `get_time` and `get_date` dictionaries (with the exception of dst = day light standard time, as it cannot be determined from epoch).

- **int** `get_dynamic_memory_usage()` const
  Return the total amount of dynamic memory used (only works in debug).

- **Dictionary** `get_engine_version()` const
  Returns the current engine version information in a Dictionary.
  “major” - Holds the major version number as a String
  “minor” - Holds the minor version number as a String
  “patch” - Holds the patch version number as a String
  “status” - Holds the status (e.g. “beta”, “rc1”, “rc2”, ... “stable”) as a String
  “revision” - Holds the revision (e.g. ”custom-build”) as a String
  “string” - major + minor + patch + status + revision in a single String

- **String** `get_environment(String environment)` const
  Return an environment variable.

- **String** `get_executable_path()` const
  Return the path to the current engine executable.

- **int** `get_exit_code()` const

- **int** `get_frames_drawn()` const
  Return the total amount of frames drawn.
- `float get_frames_per_second ( ) const`
  Returns the frames per second of the running game.

- `Array get_fullscreen_mode_list ( int screen=0 ) const`
  Return the list of fullscreen modes.

- `int get_iterations_per_second ( ) const`
  Return the amount of fixed iterations per second (for fixed process and physics).

- `String get_latin_keyboard_variant ( ) const`
  Returns the current latin keyboard variant as a String.
  Possible return values are: “QWERTY”, “AZERTY”, “QZERTY”, “DVORAK”, “NEO” or “ERROR”

- `String get_locale ( ) const`
  Return the host OS locale.

- `Object get_main_loop ( ) const`
  Return the main loop object (see MainLoop).

- `String get_model_name ( ) const`
  Returns the model name of the current device.

- `String get_name ( ) const`

- `int get_process_ID ( ) const`
  Returns the game process ID.

- `int get_processor_count ( ) const`
  Returns the number of cores available in the host machine.

- `String get_scancode_string ( int code ) const`
  Returns the given scancode as a string (e.g. Return values: “Escape”, “Shift+Escape”).

- `int get_screen_count ( ) const`
  Returns the number of displays attached to the host machine.

- `int get_screen_dpi ( int screen=0 ) const`
  Returns the dots per inch density of the specified screen.

On Android Devices, the actual screen densities are grouped into six generalized densities:

- ldpi - 120 dpi
- mdpi - 160 dpi
- hdpi - 240 dpi
- xhdpi - 320 dpi
- xxhdpi - 480 dpi
- xxxhdpi - 640 dpi

- `int get_screen_orientation ( ) const`
Returns the current screen orientation, the return value will be one of the SCREEN_ORIENTATION constants in this class.

- `Vector2 get_screen_position ( int screen=0 ) const`
- `Vector2 get_screen_size ( int screen=0 ) const`

Returns the dimensions in pixels of the specified screen.

- `int get_splash_tick_msec ( ) const`
- `int get_static_memory_peak_usage ( ) const`

Return the max amount of static memory used (only works in debug).

- `int get_static_memory_usage ( ) const`
- `String get_system_dir ( int dir ) const`
- `int get_system_time_secs ( ) const`
- `float get_target_fps ( ) const`
- `int get_ticks_msec ( ) const`

Return the amount of time passed in milliseconds since the engine started.

- `Dictionary get_time ( bool utc=false ) const`

Returns current time as a dictionary of keys: hour, minute, second

- `float get_time_scale ( )`
- `Dictionary get_time_zone_info ( ) const`
- `String get_unique_ID ( ) const`
- `int get_unix_time ( ) const`

Return the current unix timestamp.

- `int get_unix_time_from_datetime ( Dictionary datetime ) const`

Get an epoch time value from a dictionary of time values. `datetime` must be populated with the following keys: year, month, day, hour, minute, second.

You can pass the output from `get_datetime_from_unix_time` directly into this function. Daylight savings time (dst), if present, is ignored.

- `Vector2 get_video_mode_size ( int screen=0 ) const`

Return the current video mode size.

- `Vector2 get_window_position ( ) const`

Returns the window position relative to the screen, the origin is the top left corner, +Y axis goes to the bottom and +X axis goes to the right.

- `Vector2 get_window_size ( ) const`

Returns the size of the window (without counting window manager decorations).

- `bool has_environment ( String environment ) const`

Return true if an environment variable exists.

- `bool has_touchscreen_ui_hint ( ) const`
- `bool has_virtual_keyboard ( ) const`
Returns true if the platform has a virtual keyboard, false otherwise.

- void hide_virtual_keyboard()

Hides the virtual keyboard if it is shown, does nothing otherwise.

- bool is_debug_build() const
- bool is_in_low_processor_usage_mode() const

Return true if low cpu usage mode is enabled.

- bool is_keep_screen_on() const

Returns whether the screen is being kept on or not.

- bool is_ok_left_and_cancel_right() const
- bool is_scancode_unicode(int code) const
- bool is_stdout_verbose() const

Return true if the engine was executed with -v (verbose stdout).

- bool is_video_mode_fullscreen(int screen=0) const

Return true if the current video mode is fullscreen.

- bool is_video_mode_resizable(int screen=0) const

Return true if the window is resizable.

- bool is_vsync_enabled() const
- bool is_window_fullscreen() const

Returns whether the window is in fullscreen mode or not.

- bool is_window_maximized() const

Return true if the window is maximized.

- bool is_window_minimized() const

Return true if the window is minimized.

- bool is_window_resizable() const

Returns whether the window is resizable or not.

- int kill(int pid)

Kill a process ID (this method can be used to kill processes that were not spawned by the game).

- bool native_video_is_playing()

- void native_video_pause()

- int native_video_play(String path, float volume, String audio_track, String subtitle_track)

- void native_video_stop()

- void native_video_unpause()

- void print_all_resources(String tofile="")

- void print_all_textures_by_size()

- void print_resources_by_type(StringArray types)

- void print_resources_in_use(bool short=false)
- void request_attention ()

Request the user attention to the window. It’ll flash the taskbar button on Windows or bounce the dock icon on OSX.

- void set_borderless_window ( bool borderless )
- void set_clipboard ( String clipboard )

Set clipboard to the OS.

- void set_current_screen ( int screen )
- void set_exit_code ( int code )
- void set_icon ( Image icon )
- void set_iterations_per_second ( int iterations_per_second )

Set the amount of fixed iterations per second (for fixed process and physics).

- void set_keep_screen_on ( bool enabled )

Set keep screen on if true, or goes to sleep by device setting if false. (for Android/iOS)

- void set_low_processor_usage_mode ( bool enable )

Set to true to enable the low cpu usage mode. In this mode, the screen only redraws when there are changes, and a considerable sleep time is inserted between frames. This way, editors using the engine UI only use very little cpu.

- void set_screen_orientation ( int orientation )

Sets the current screen orientation, the argument value must be one of the SCREEN_ORIENTATION constants in this class.

- void set_target_fps ( int target_fps )

int set_thread_name ( String name )

- void set_time_scale ( float time_scale )

Speeds up or slows down the physics by changing the delta variable. (delta * time_scale)

- void set_use_file_access_save_and_swap ( bool enabled )
- void set_use_vsync ( bool enable )
- void set_video_mode ( Vector2 size, bool fullscreen, bool resizable, int screen=0 )

Change the video mode.

- void set_window_fullscreen ( bool enabled )

Sets window fullscreen mode to the enabled argument, enabled is a toggle for the fullscreen mode, calling the function with enabled true when the screen is not on fullscreen mode will cause the screen to go to fullscreen mode, calling the function with enabled false when the screen is in fullscreen mode will cause the window to exit the fullscreen mode.

- void set_window_maximized ( bool enabled )

Set the window size to maximized.

- void set_window_minimized ( bool enabled )

Set whether the window is minimized.

- void set_window_position ( Vector2 position )

Sets the position of the window to the specified position (this function could be restricted by the window manager, meaning that there could be some unreachable areas of the screen).

- void set_window_resizable ( bool enabled )
Set the window resizable state, if the window is not resizable it will preserve the dimensions specified in the project settings.

- void set_window_size (Vector2 size)
  Sets the window size to the specified size.
- void set_window_title (String title)
  Sets the window title to the specified string.
- int shell_open (String uri)
- void show_virtual_keyboard (String existing_text="")
  Shows the virtual keyboard if the platform has one. The existing_text parameter is useful for implementing your own LineEdit, as it tells the virtual keyboard what text has already been typed (the virtual keyboard uses it for auto-correct and predictions).

### 12.188 PackedDataContainer

Inherits: Resource < Reference < Object

Category: Core

#### 12.188.1 Brief Description

#### 12.188.2 Member Functions

<table>
<thead>
<tr>
<th>Error</th>
<th>pack (Variant value)</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>size () const</td>
</tr>
</tbody>
</table>

#### 12.188.3 Member Function Description

- Error pack (Variant value)
- int size () const

### 12.189 PackedDataContainerRef

Inherits: Reference < Object

Category: Core

#### 12.189.1 Brief Description

#### 12.189.2 Member Functions

<table>
<thead>
<tr>
<th>int</th>
<th>size () const</th>
</tr>
</thead>
</table>
12.189.3 Member Function Description

- int size ( ) const

12.190 PackedScene

Inherits: Resource < Reference < Object
Category: Core

12.190.1 Brief Description

12.190.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool</td>
<td>can_instance ( ) const</td>
<td></td>
</tr>
<tr>
<td>SceneState</td>
<td>get_state ( )</td>
<td></td>
</tr>
<tr>
<td>Node</td>
<td>instance ( bool gen_edit_state=false ) const</td>
<td></td>
</tr>
<tr>
<td>int</td>
<td>pack ( Node path )</td>
<td></td>
</tr>
</tbody>
</table>

12.190.3 Description

TODO: explain ownership, and that node does not need to own itself

12.190.4 Member Function Description

- bool can_instance ( ) const
- SceneState get_state ( )
- Node instance ( bool gen_edit_state=false ) const
- int pack ( Node path )

Pack will ignore any sub-nodes not owned by given node. See Node.set_owner.

12.191 PacketPeer

Inherits: Reference < Object
Inherited By: PacketPeerStream, PacketPeerUDP
Category: Core

12.191.1 Brief Description

Abstraction and base class for packet-based protocols.
12.191.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>get_available_packet_count() const</td>
</tr>
<tr>
<td>RawArray</td>
<td>get_packet() const</td>
</tr>
<tr>
<td>Error</td>
<td>get_packet_error() const</td>
</tr>
<tr>
<td>Variant</td>
<td>get_var() const</td>
</tr>
<tr>
<td>Error</td>
<td>put_packet(RawArray buffer)</td>
</tr>
<tr>
<td>int</td>
<td>put_var(Variant var)</td>
</tr>
</tbody>
</table>

12.191.3 Description

PacketPeer is an abstraction and base class for packet-based protocols (such as UDP). It provides an API for sending and receiving packets both as raw data or variables. This makes it easy to transfer data over a protocol, without having to encode data as low level bytes or having to worry about network ordering.

12.191.4 Member Function Description

- **int get_available_packet_count() const**
  
  Return the number of packets currently available in the ring-buffer.

- **RawArray get_packet() const**
  
  Get a raw packet.

- **Error get_packet_error() const**
  
  Return the error state of the last packet received (via `get_packet` and `get_var`).

- **Variant get_var() const**
  
  Get a Variant.

- **Error put_packet(RawArray buffer)**
  
  Send a raw packet.

- **int put_var(Variant var)**
  
  Send a Variant as a packet.

12.192 PacketPeerStream

**Inherits:** PacketPeer < Reference < Object

**Category:** Core

12.192.1 Brief Description

Wrapper to use a PacketPeer over a StreamPeer.

12.192.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>set_stream_peer(StreamPeer peer)</td>
</tr>
</tbody>
</table>
12.192.3 Description

PacketStreamPeer provides a wrapper for working using packets over a stream. This allows for using packet based code with StreamPeers. PacketPeerStream implements a custom protocol over the StreamPeer, so the user should not read or write to the wrapped StreamPeer directly.

12.192.4 Member Function Description

- void set_stream_peer (StreamPeer peer)
  Set the StreamPeer object to be wrapped

12.193 PacketPeerUDP

Inherits: PacketPeer < Reference < Object

Category: Core

12.193.1 Brief Description

UDP packet peer.

12.193.2 Member Functions

<table>
<thead>
<tr>
<th>void</th>
<th>close ( )</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>get_packet_ip ( ) const</td>
</tr>
<tr>
<td>int</td>
<td>get_packet_port ( ) const</td>
</tr>
<tr>
<td>bool</td>
<td>is_listening ( ) const</td>
</tr>
<tr>
<td>Error</td>
<td>listen ( int port, String bind_address=&quot;*&quot;, int recv_buf_size=65536 )</td>
</tr>
<tr>
<td>int</td>
<td>set_send_address ( String host, int port )</td>
</tr>
<tr>
<td>Error</td>
<td>wait ( )</td>
</tr>
</tbody>
</table>

12.193.3 Description

UDP packet peer. Can be used to send raw UDP packets as well as Variants.

12.193.4 Member Function Description

- void close ( )
  Close the UDP socket the PacketPeerUDP is currently listening on.
  - String get_packet_ip ( ) const
  Return the IP of the remote peer that sent the last packet(that was received with get_packet or get_var).
  - int get_packet_port ( ) const
  Return the port of the remote peer that sent the last packet(that was received with get_packet or get_var).
  - bool is_listening ( ) const
Return whether this *PacketPeerUDP* is listening.

- Error `listen ( int port, String bind_address="*", int recv_buf_size=65536 )`

Make this *PacketPeerUDP* listen on the “port” binding to “bind_address” with a buffer size “recv_buf_size”.

If “bind_address” is set as “*” (default), the peer will listen on all available addresses (both IPv4 and IPv6).

If “bind_address” is set as “0.0.0.0” (for IPv4) or “::” (for IPv6), the peer will listen on all available addresses matching that IP type.

If “bind_address” is set to any valid address (e.g. “192.168.1.101”, “::1”, etc), the peer will only listen on the interface with that addresses (or fail if no interface with the given address exists).

- `int set_send_address ( String host, int port )`

Set the destination address and port for sending packets and variables, a hostname will be resolved using if valid.

- Error `wait ( )`

Wait for a packet to arrive on the listening port, see *listen*.

### 12.194 Panel

**Inherits:** *Control < CanvasItem < Node < Object*

**Category:** Core

#### 12.194.1 Brief Description

Provides an opaque background for *Control* children.

#### 12.194.2 Description

Panel is a *Control* that displays an opaque background. It’s commonly used as a parent and container for other types of *Control* nodes.

### 12.195 PanelContainer

**Inherits:** *Container < Control < CanvasItem < Node < Object*

**Category:** Core

#### 12.195.1 Brief Description

Panel container type.

#### 12.195.2 Description

Panel container type. This container fits controls inside of the delimited area of a stylebox. It’s useful for giving controls an outline.
12.196 ParallaxBackground

Inherits: CanvasLayer < Node < Object
Category: Core

12.196.1 Brief Description

A node used to create a parallax scrolling background.

12.196.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vector2</td>
<td>get_limit_begin ( ) const</td>
<td>Return the beginning limit.</td>
</tr>
<tr>
<td>Vector2</td>
<td>get_limit_end ( ) const</td>
<td>Return the ending limit.</td>
</tr>
<tr>
<td>Vector2</td>
<td>get_scroll_base_offset ( ) const</td>
<td>Return the base offset.</td>
</tr>
<tr>
<td>Vector2</td>
<td>get_scroll_base_scale ( ) const</td>
<td>Return the base motion scale.</td>
</tr>
<tr>
<td>Vector2</td>
<td>get_scroll_offset ( ) const</td>
<td></td>
</tr>
<tr>
<td>bool</td>
<td>is_ignore_camera_zoom ( )</td>
<td>Return ignoring camera zoom.</td>
</tr>
<tr>
<td>void</td>
<td>set_ignore_camera_zoom ( bool ignore )</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>set_limit_begin ( Vector2 ofs )</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>set_limit_end ( Vector2 ofs )</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>set_scroll_base_offset ( Vector2 ofs )</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>set_scroll_base_scale ( Vector2 scale )</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>set_scroll_offset ( Vector2 ofs )</td>
<td></td>
</tr>
</tbody>
</table>

12.196.3 Description

A ParallaxBackground will use one or more ParallaxLayer nodes to create a parallax scrolling background. Each ParallaxLayer can be set to move at different speeds relative to the camera movement, this can be used to create an illusion of depth in a 2D game.

12.196.4 Member Function Description

- **Vector2 get_limit_begin ( ) const**
  Return the beginning limit.

- **Vector2 get_limit_end ( ) const**
  Return the ending limit.

- **Vector2 get_scroll_base_offset ( ) const**
  Return the base offset.

- **Vector2 get_scroll_base_scale ( ) const**
  Return the base motion scale.

- **Vector2 get_scroll_offset ( ) const**

- **bool is_ignore_camera_zoom ( )**
  Return ignoring camera zoom.
• void **set_ignore_camera_zoom** ( bool ignore )
Set to true for all child **ParallaxLayer** nodes to not be affected by the zoom level of the camera.

• void **set_limit_begin** ( Vector2 ofs )
Set the left and top limits in pixels for scrolling to begin. If the camera is outside of this limit the background will not continue to scroll. If an axis is greater than or equal to the corresponding axis of limit_end, then it will not limit scrolling for that axis.

• void **set_limit_end** ( Vector2 ofs )
Set the right and bottom limits in pixels for scrolling to end. If the camera is outside of this limit the background will not continue to scroll. If an axis is less than or equal to the corresponding axis of limit_begin, then it will not limit scrolling for that axis.

• void **set_scroll_base_offset** ( Vector2 ofs )
Set the base offset in pixels of all children **ParallaxLayer** nodes.

• void **set_scroll_base_scale** ( Vector2 scale )
Set the base motion scale of all children **ParallaxLayer** nodes.

• void **set_scroll_offset** ( Vector2 ofs )

### 12.197 ParallaxLayer

**Inherits:** Node2D < CanvasItem < Node < Object

**Category:** Core

#### 12.197.1 Brief Description

A parallax scrolling layer to be used with **ParallaxBackground**.

#### 12.197.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vector2</td>
<td>get_mirroring () const</td>
</tr>
<tr>
<td>Vector2</td>
<td>get_motion_offset () const</td>
</tr>
<tr>
<td>Vector2</td>
<td>get_motion_scale () const</td>
</tr>
<tr>
<td>void</td>
<td>set_mirroring ( Vector2 mirror )</td>
</tr>
<tr>
<td>void</td>
<td>set_motion_offset ( Vector2 offset )</td>
</tr>
<tr>
<td>void</td>
<td>set_motion_scale ( Vector2 scale )</td>
</tr>
</tbody>
</table>

#### 12.197.3 Description

A **ParallaxLayer** must be the child of a **ParallaxBackground** node. All child nodes will be affected by the parallax scrolling of this layer.
12.197.4 Member Function Description

- `Vector2 get_mirroring ( ) const`
  Return the mirroring of the ParallaxLayer.
  - `Vector2 get_motion_offset ( ) const`
  - `Vector2 get_motion_scale ( ) const`
  Return the motion scale of the ParallaxLayer.
- `void set_mirroring ( Vector2 mirror )`
  Set the mirroring of the ParallaxLayer. If an axis is set to 0 then that axis will have no mirroring.
- `void set_motion_offset ( Vector2 offset )`
- `void set_motion_scale ( Vector2 scale )`
  Set the motion scale of the ParallaxLayer. If an axis is set to 0 then it will not move at all, it will stick with the camera.

12.198 ParticleAttractor2D

Inherits: Node2D < CanvasItem < Node < Object

Category: Core

12.198.1 Brief Description

12.198.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>float</td>
<td>get_absorption ( ) const</td>
</tr>
<tr>
<td>float</td>
<td>get_disable_radius ( ) const</td>
</tr>
<tr>
<td>float</td>
<td>get_gravity ( ) const</td>
</tr>
<tr>
<td>NodePath</td>
<td>get_particles_path ( ) const</td>
</tr>
<tr>
<td>float</td>
<td>get_radius ( ) const</td>
</tr>
<tr>
<td>bool</td>
<td>is_enabled ( ) const</td>
</tr>
<tr>
<td>void</td>
<td>set_absorption ( float absorption )</td>
</tr>
<tr>
<td>void</td>
<td>set_disable_radius ( float radius )</td>
</tr>
<tr>
<td>void</td>
<td>set_enabled ( bool enabled )</td>
</tr>
<tr>
<td>void</td>
<td>set_gravity ( float gravity )</td>
</tr>
<tr>
<td>void</td>
<td>set_particles_path ( NodePath path )</td>
</tr>
<tr>
<td>void</td>
<td>set_radius ( float radius )</td>
</tr>
</tbody>
</table>

12.198.3 Member Function Description

- `float get_absorption ( ) const`
- `float get_disable_radius ( ) const`
- `float get_gravity ( ) const`
- `NodePath get_particles_path ( ) const`
- `float get_radius ( ) const`
12.199 Particles

Inherits: GeometryInstance < VisualInstance < Spatial < Node < Object

Category: Core

12.199.1 Brief Description

Particle system 3D Node

12.199.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>get_amount() const</td>
</tr>
<tr>
<td>Color</td>
<td>get_color_phase_color() const (int phase)</td>
</tr>
<tr>
<td>float</td>
<td>get_color_phase_pos() const (int phase)</td>
</tr>
<tr>
<td>int</td>
<td>get_color_phases() const</td>
</tr>
<tr>
<td>Vector3</td>
<td>get_emission_base_velocity() const</td>
</tr>
<tr>
<td>Vector3</td>
<td>get_emission_half_extents() const</td>
</tr>
<tr>
<td>Vector3Array</td>
<td>get_emission_points() const</td>
</tr>
<tr>
<td>float</td>
<td>get_emit_timeout() const</td>
</tr>
<tr>
<td>Vector3</td>
<td>get_gravity_normal() const</td>
</tr>
<tr>
<td>Material</td>
<td>get_material() const</td>
</tr>
<tr>
<td>float</td>
<td>get_randomness() const (int variable)</td>
</tr>
<tr>
<td>float</td>
<td>get_variable() const (int variable)</td>
</tr>
<tr>
<td>AABB</td>
<td>get_visibility_aabb() const</td>
</tr>
<tr>
<td>bool</td>
<td>has_height_from_velocity() const</td>
</tr>
<tr>
<td>bool</td>
<td>is_emitting() const</td>
</tr>
<tr>
<td>bool</td>
<td>is_using_local_coordinates() const</td>
</tr>
<tr>
<td>void</td>
<td>set_amount(int amount)</td>
</tr>
<tr>
<td>void</td>
<td>set_color_phase_color(int phase, Color color)</td>
</tr>
<tr>
<td>void</td>
<td>set_color_phase_pos(int phase, float pos)</td>
</tr>
<tr>
<td>void</td>
<td>set_color_phases(int count)</td>
</tr>
<tr>
<td>void</td>
<td>set_emission_base_velocity(Vector3 base_velocity)</td>
</tr>
<tr>
<td>void</td>
<td>set_emission_half_extents(Vector3 half_extents)</td>
</tr>
<tr>
<td>void</td>
<td>set_emission_points(Vector3Array points)</td>
</tr>
<tr>
<td>void</td>
<td>set_emit_timeout(float timeout)</td>
</tr>
<tr>
<td>void</td>
<td>set_emitting(bool enabled)</td>
</tr>
</tbody>
</table>

Continued on next page
Table 12.24 – continued from previous page

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void set_gravity_normal(Vector3 normal)</code></td>
<td>Set the gravity vector to a normal direction.</td>
</tr>
<tr>
<td><code>void set_height_from_velocity(bool enable)</code></td>
<td>Set the height based on velocity.</td>
</tr>
<tr>
<td><code>void set_material(Material material)</code></td>
<td>Set the material of the particle.</td>
</tr>
<tr>
<td><code>void set_randomness(int variable, float randomness)</code></td>
<td>Set randomness.</td>
</tr>
<tr>
<td><code>void set_use_local_coordinates(bool enable)</code></td>
<td>Use local coordinates.</td>
</tr>
<tr>
<td><code>void set_variable(int variable, float value)</code></td>
<td>Set a variable with the given value.</td>
</tr>
<tr>
<td><code>void set_visibility_aabb(AABB aabb)</code></td>
<td>Set visibility AABB.</td>
</tr>
</tbody>
</table>

12.199.3 Numeric Constants

- VAR_LIFETIME = 0
- VAR_SPREAD = 1
- VAR_GRAVITY = 2
- VAR_LINEAR_VELOCITY = 3
- VAR_ANGULAR_VELOCITY = 4
- VAR_LINEAR_ACCELERATION = 5
- VAR_DRAG = 6
- VAR_TANGENTIAL_ACCELERATION = 7
- VAR_INITIAL_SIZE = 9
- VAR_FINAL_SIZE = 10
- VAR_INITIAL_ANGLE = 11
- VAR_HEIGHT = 12
- VAR_HEIGHT_SPEED_SCALE = 13
- VAR_MAX = 14

12.199.4 Description

Particles is a particle system 3D Node that is used to simulate several types of particle effects, such as explosions, rain, snow, fireflies, or other magical-like shiny sparkles. Particles are drawn using impostors, and given their dynamic behavior, the user must provide a visibility AABB (although helpers to create one automatically exist).

12.199.5 Member Function Description

- int get_amount() const
  Return the total amount of particles in the system.

- Color get_color_phase_color(int phase) const
  Return the color of a color phase.

- float get_color_phase_pos(int phase) const
  Return the position of a color phase (0 to 1).
• \texttt{Vector3 get\_emission\_base\_velocity()} const
• \texttt{Vector3 get\_emission\_half\_extents()} const

Return the half extents for the emission box.

• \texttt{Vector3Array get\_emission\_points()} const
• \texttt{float get\_emit\_timeout()} const
• \texttt{Vector3 get\_gravity\_normal()} const

Return the normal vector towards where gravity is pulling (by default, negative Y).

• \texttt{Material get\_material()} const

Return the material used to draw particles.

• \texttt{float get\_randomness(int variable)} const

Return the randomness for a specific variable of the particle system. Randomness produces small changes from the default each time a particle is emitted.

• \texttt{float get\_variable(int variable)} const

Return a specific variable for the particle system (see \texttt{VAR_*} enum).

• \texttt{AABB get\_visibility\_aabb()} const

Return the current visibility AABB.

• \texttt{bool has\_height\_from\_velocity()} const
• \texttt{bool is\_emitting()} const

Return the “emitting” property state (see \texttt{set\_emitting}).

• \texttt{bool is\_using\_local\_coordinates()} const
• \texttt{void set\_amount(int amount)}

Set total amount of particles in the system.

• \texttt{void set\_color\_phase\_color(int phase, Color color)}

Set the color of a color phase.

• \texttt{void set\_color\_phase\_pos(int phase, float pos)}

Set the position of a color phase (0 to 1).

• \texttt{void set\_color\_phases(int count)}
• \texttt{void set\_emission\_base\_velocity(Vector3 base\_velocity)}
• \texttt{void set\_emission\_half\_extents(Vector3 half\_extents)}

Set the half extents for the emission box.

• \texttt{void set\_emission\_points(Vector3Array points)}
• \texttt{void set\_emit\_timeout(float timeout)}
• \texttt{void set\_emitting(bool enabled)}

Set the “emitting” property state. When emitting, the particle system generates new particles at constant rate.

• \texttt{void set\_gravity\_normal(Vector3 normal)}

Set the normal vector towards where gravity is pulling (by default, negative Y).
• void set_height_from_velocity ( bool enable )
• void set_material ( Material material )
Set the material used to draw particles.

• void set_randomness ( int variable, float randomness )
Set the randomness for a specific variable of the particle system. Randomness produces small changes from the default each time a particle is emitted.

• void set_use_local_coordinates ( bool enable )
• void set_variable ( int variable, float value )
Set a specific variable for the particle system (see VAR_* enum).

• void set_visibility_aabb ( AABB aabb )
Set the visibility AABB for the particle system, since the default one will not work properly most of the time.

12.200 Particles2D

Inherits: Node2D < CanvasItem < Node < Object

Category: Core

12.200.1 Brief Description

2D Particle emitter

12.200.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>get_amount ( ) const</td>
</tr>
<tr>
<td>Color</td>
<td>get_color ( ) const</td>
</tr>
<tr>
<td>Color</td>
<td>get_color_phase_color ( int phase ) const</td>
</tr>
<tr>
<td>float</td>
<td>get_color_phase_pos ( int phase ) const</td>
</tr>
<tr>
<td>int</td>
<td>get_color_phases ( ) const</td>
</tr>
<tr>
<td>ColorRamp</td>
<td>get_color_ramp ( ) const</td>
</tr>
<tr>
<td>Vector2</td>
<td>get_emission_half_extents ( ) const</td>
</tr>
<tr>
<td>Vector2Array</td>
<td>get_emission_points ( ) const</td>
</tr>
<tr>
<td>Vector2</td>
<td>get_emissoor_offset ( ) const</td>
</tr>
<tr>
<td>float</td>
<td>get_emit_timeout ( ) const</td>
</tr>
<tr>
<td>float</td>
<td>get_explosiveness ( ) const</td>
</tr>
<tr>
<td>int</td>
<td>get_h_frames ( ) const</td>
</tr>
<tr>
<td>Vector2</td>
<td>get_initial_velocity ( ) const</td>
</tr>
<tr>
<td>float</td>
<td>get_lifetime ( ) const</td>
</tr>
<tr>
<td>float</td>
<td>get_param ( int param ) const</td>
</tr>
<tr>
<td>float</td>
<td>get_pre_process_time ( ) const</td>
</tr>
<tr>
<td>int</td>
<td>get_process_mode ( ) const</td>
</tr>
<tr>
<td>float</td>
<td>get_randomness ( int param ) const</td>
</tr>
<tr>
<td>Texture</td>
<td>get_texture ( ) const</td>
</tr>
<tr>
<td>float</td>
<td>get_time_scale ( ) const</td>
</tr>
</tbody>
</table>

Continued on next page
12.200.3 Numeric Constants

- **PARAM_DIRECTION = 0** — Direction in degrees at which the particles will be launched. Notice that when the direction is set to 0 the particles will be launched to the negative.

- **PARAM_SPREAD = 1**

- **PARAM_LINEAR VELOCITY = 2** — Velocity at which the particles will be launched.

- **PARAM_SPIN VELOCITY = 3** — The speed at which particles will spin around its own center.

- **PARAM_ORBIT VELOCITY = 4** — Velocity at which the particles will orbit around the emitter center.

- **PARAM_GRAVITY_DIRECTION = 5** — Direction in degrees at which the particles will be attracted.

- **PARAM_GRAVITY_STRENGTH = 6** — Strength of the gravitation attraction for each particle.

- **PARAM_RADIAL ACCEL = 7**

- **PARAM_TANGENTIAL_ACCEL = 8**

- **PARAM_DAMPING = 9** — Amount of damping for each particle.
• **PARAM\_INITIAL\_ANGLE** = 10 — Initial angle in radians at which each particle will be spawned
• **PARAM\_INITIAL\_SIZE** = 11 — Initial size of each particle
• **PARAM\_FINAL\_SIZE** = 12 — Final size of each particle, the particle size will interpolate to this value during its lifetime.
• **PARAM\_HUE\_VARIATION** = 13
• **PARAM\_ANIM\_SPEED\_SCALE** = 14
• **PARAM\_ANIM\_INITIAL\_POS** = 15
• **PARAM\_MAX** = 16
• **MAX\_COLOR\_PHASES** = 4

### 12.200.4 Description

Particles2D is a particle system 2D Node that is used to simulate several types of particle effects, such as explosions, rain, snow, fireflies, or other magical-like shiny sparkles. Particles are drawn using impostors, and given their dynamic behavior, the user must provide a visibility AABB (although helpers to create one automatically exist).

### 12.200.5 Member Function Description

- **int get_amount ( ) const**
  Returns the amount of particles spawned at each emission

- **Color get_color ( ) const**
  Returns the tint color for each particle.

- **Color get_color_phase_color ( int phase ) const**

- **float get_color_phase_pos ( int phase ) const**

- **int get_color_phases ( ) const**

- **ColorRamp get_color_ramp ( ) const**
  Returns the ColorRamp used to tint each particle

- **Vector2 get_emission_half_extents ( ) const**
  Returns the half extents of the emission box.

- **Vector2Array get_emission_points ( ) const**

- **Vector2 get_emissor_offset ( ) const**
  Returns the particle spawn origin position relative to the emitter.

- **float get_emit_timeout ( ) const**
  Returns the amount of seconds during which the emitter will spawn particles

- **float get_explosiveness ( ) const**

- **int get_h_frames ( ) const**

- **Vector2 get_initial_velocity ( ) const**

- **float get_lifetime ( ) const**
  Gets the amount of seconds that each particle will be visible.
- `float get_param (int param) const`

Returns the value of the specified emitter parameter

- `float get_pre_process_time () const`
- `int get_process_mode () const`
- `float get_randomness (int param) const`

Returns the randomness value of the specified emitter parameter

- `Texture get_texture () const`

Returns the texture for emitted particles

- `float get_time_scale () const`

Returns the emitter time scale

- `int get_v_frames () const`
- `bool is_emitting () const`

Returns whether this emitter is currently emitting or not

- `bool is_flipped_h () const`
- `bool is_flipped_v () const`
- `bool is_using_local_space () const`
- `void pre_process (float time)`
- `void reset ()`
- `void set_amount (int amount)`

Sets the amount of particles spawned at each emission

- `void set_color (Color color)`

Set the tint color for each particle.

- `void set_color_phase_color (int phase, Color color)`
- `void set_color_phase_pos (int phase, float pos)`
- `void set_color_phases (int phases)`
- `ColorRamp set_color_ramp (Object color_ramp)`

Sets the `ColorRamp` used to tint each particle. Particle will be tinted according to their lifetimes.

- `void set_emission_half_extents (Vector2 extents)`

Sets the half extents of the emission box, particles will be spawned at random inside this box.

- `void set_emission_points (Vector2Array points)`
- `void set_emissor_offset (Vector2 offset)`

Sets the particle spawn origin position relative to the emitter center. For example if this value is set to (50, 50), the particle will spawn 50 units to the right and 50 units to the bottom of the emitter center.

- `void set_emit_timeout (float value)`

Sets the amount of seconds during which the emitter will spawn particles, after the specified seconds the emitter state will be set to non emitting, so calling `is_emitting` will return false. If the timeout is 0 the emitter will spawn forever.

- `void set_emitting (bool active)`
If this is set to true then the particle emitter will emit particles, if its false it will not.

- void set_explosiveness (float amount)
- void set_flip_h (bool enable)
- void set_flip_v (bool enable)
- void set_h_frames (int enable)
- void set_initial_velocity (Vector2 velocity)
- void set_lifetime (float lifetime)

Sets the amount of seconds that each particle will be visible.

- void set_param (int param, float value)

Sets the value of the specified emitter parameter (see the constants section for the list of parameters)

- void set_pre_process_time (float time)
- void set_process_mode (int mode)
- void set_randomness (int param, float value)

Sets the randomness value of the specified emitter parameter (see the constants section for the list of parameters), 0 means no randomness, so every particle will have the parameters specified, 1 means that the parameter will be chosen at random, the closer the randomness value gets to 0 the more conservative the variation of the parameter will be.

- Texture set_texture (Object texture)

Sets the texture for each particle

- void set_time_scale (float time_scale)

Sets the increment or decrement for the particle lifetime. for example: if the time scale is set to 2, the particles will die and move twice as fast.

- void set_use_local_space (bool enable)
- void set_v_frames (int enable)

### 12.201 Patch9Frame

Inherits: Control < CanvasItem < Node < Object

Category: Core
12.201.1 Brief Description

12.201.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool</td>
<td><code>get_draw_center()</code> const</td>
</tr>
<tr>
<td>Color</td>
<td><code>get_modulate()</code> const</td>
</tr>
<tr>
<td>int</td>
<td><code>get_patch_margin(int margin)</code> const</td>
</tr>
<tr>
<td>Rect2</td>
<td><code>get_region_rect()</code> const</td>
</tr>
<tr>
<td>Object</td>
<td><code>get_texture()</code> const</td>
</tr>
<tr>
<td>void</td>
<td><code>set_draw_center(bool draw_center)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_modulate(Color modulate)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_patch_margin(int margin, int value)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_region_rect(Rect2 rect)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_texture(Object texture)</code></td>
</tr>
</tbody>
</table>

12.201.3 Signals

- `texture_changed()`

12.201.4 Member Function Description

- `bool get_draw_center()` const
- `Color get_modulate()` const
- `int get_patch_margin(int margin)` const
- `Rect2 get_region_rect()` const
- `Object get_texture()` const
- `void set_draw_center(bool draw_center)`
- `void set_modulate(Color modulate)`
- `void set_patch_margin(int margin, int value)`
- `void set_region_rect(Rect2 rect)`
- `void set_texture(Object texture)`

12.202 Path

Inherits: `Spatial < Node < Object`

Category: Core

12.202.1 Brief Description

Container for a `Curve3D`.
12.202.2 Member Functions

<table>
<thead>
<tr>
<th>Curve3D</th>
<th>get_curve ( ) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>set_curve ( Curve3D curve )</td>
</tr>
</tbody>
</table>

12.202.3 Description

This class is a container/Node-ification of a Curve3D, so it can have Spatial properties and Node info.

12.202.4 Member Function Description

- *Curve3D* get_curve ( ) const
  Returns the Curve3D contained.
- void set_curve ( Curve3D curve )
  Sets the Curve3D.

12.203 Path2D

Inherits: Node2D < CanvasItem < Node < Object

Category: Core

12.203.1 Brief Description

Container for a Curve2D.

12.203.2 Member Functions

<table>
<thead>
<tr>
<th>Curve2D</th>
<th>get_curve ( ) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>set_curve ( Curve2D curve )</td>
</tr>
</tbody>
</table>

12.203.3 Description

This class is a container/Node-ification of a Curve2D, so it can have Node2D properties and Node info.

12.203.4 Member Function Description

- *Curve2D* get_curve ( ) const
  Returns the Curve2D contained.
- void set_curve ( Curve2D curve )
  Sets the Curve2D.
12.204 PathFollow

Inherits: Spatial < Node < Object

Category: Core

12.204.1 Brief Description

Point sampler for a Path.

12.204.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool</td>
<td>get_cubic_interpolation ( ) const</td>
</tr>
<tr>
<td>float</td>
<td>get_h_offset ( ) const</td>
</tr>
<tr>
<td>float</td>
<td>get_offset ( ) const</td>
</tr>
<tr>
<td>int</td>
<td>get_rotation_mode ( ) const</td>
</tr>
<tr>
<td>float</td>
<td>get_unit_offset ( ) const</td>
</tr>
<tr>
<td>float</td>
<td>get_v_offset ( ) const</td>
</tr>
<tr>
<td>bool</td>
<td>has_loop ( ) const</td>
</tr>
<tr>
<td>void</td>
<td>set_cubic_interpolation ( bool enable )</td>
</tr>
<tr>
<td>void</td>
<td>set_h_offset ( float h_offset )</td>
</tr>
<tr>
<td>void</td>
<td>set_loop ( bool loop )</td>
</tr>
<tr>
<td>void</td>
<td>set_offset ( float offset )</td>
</tr>
<tr>
<td>void</td>
<td>set_rotation_mode ( int rotation_mode )</td>
</tr>
<tr>
<td>void</td>
<td>set_unit_offset ( float unit_offset )</td>
</tr>
<tr>
<td>void</td>
<td>set_v_offset ( float v_offset )</td>
</tr>
</tbody>
</table>

12.204.3 Numeric Constants

- ROTATION_NONE = 0 — Forbids the PathFollow to rotate.
- ROTATION_Y = 1 — Allows the PathFollow to rotate in the Y axis only.
- ROTATION_XY = 2 — Allows the PathFollow to rotate in both the X, and Y axes.
- ROTATION_XYZ = 3 — Allows the PathFollow to rotate in any axis.

12.204.4 Description

This node takes its parent Path, and returns the coordinates of a point within it, given a distance from the first vertex. It is useful for making other nodes follow a path, without coding the movement pattern. For that, the nodes must be descendants of this node. Then, when setting an offset in this node, the descendant nodes will move accordingly.

12.204.5 Member Function Description

- bool get_cubic_interpolation ( ) const
  This method returns whether the position between two cached points (see set_cubic_interpolation) is interpolated linearly, or cubically.

- float get_h_offset ( ) const
Returns the X displacement this node has from its parent Path.

- `float get_offset()` const

Returns the distance along the path in 3D units.

- `int get_rotation_mode()` const

Returns the rotation mode. The constants below list which axes are allowed to rotate for each mode.

- `float get_unit_offset()` const

Returns the distance along the path as a number in the range 0.0 (for the first vertex) to 1.0 (for the last).

- `float get_v_offset()` const

Returns the Y displacement this node has from its parent Path.

- `bool has_loop()` const

Returns whether this node wraps its offsets around, or truncates them to the path ends.

- `void set_cubic_interpolation(bool enable)`

The points along the Curve3D of the Path are precomputed before use, for faster calculations. The point at the requested offset is then calculated interpolating between two adjacent cached points. This may present a problem if the curve makes sharp turns, as the cached points may not follow the curve closely enough.

There are two answers to this problem: Either increase the number of cached points and increase memory consumption, or make a cubic interpolation between two points at the cost of (slightly) slower calculations.

This method controls whether the position between two cached points is interpolated linearly, or cubicly.

- `void set_h_offset(float h_offset)`

Moves this node in the X axis. As this node’s position will be set every time its offset is set, this allows many PathFollow to share the same curve (and thus the same movement pattern), yet not return the same position for a given path offset.

A similar effect may be achieved moving the this node’s descendants.

- `void set_loop(bool loop)`

If set, any offset outside the path’s length (whether set by `set_offset` or `set_unit_offset` will wrap around, instead of stopping at the ends. Set it for cyclic paths.

- `void set_offset(float offset)`

Sets the distance from the first vertex, measured in 3D units along the path. This sets this node’s position to a point within the path.

- `void set_rotation_mode(int rotation_mode)`

Allows or forbids rotation on one or more axes, per the constants below.

- `void set_unit_offset(float unit_offset)`

Sets the distance from the first vertex, considering 0.0 as the first vertex and 1.0 as the last. This is just another way of expressing the offset within the path, as the offset supplied is multiplied internally by the path’s length.

- `void set_v_offset(float v_offset)`

Moves this node in the Y axis, for the same reasons of `set_h_offset`.
12.205 PathFollow2D

Inherits: 

Category: Core

12.205.1 Brief Description

Point sampler for a Path2D.

12.205.2 Member Functions

<table>
<thead>
<tr>
<th>bool</th>
<th>get_cubic_interpolation ( ) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>float</td>
<td>get_h_offset ( ) const</td>
</tr>
<tr>
<td>float</td>
<td>get_offset ( ) const</td>
</tr>
<tr>
<td>float</td>
<td>get_unit_offset ( ) const</td>
</tr>
<tr>
<td>float</td>
<td>get_v_offset ( ) const</td>
</tr>
<tr>
<td>bool</td>
<td>has_loop ( ) const</td>
</tr>
<tr>
<td>bool</td>
<td>is_rotating ( ) const</td>
</tr>
<tr>
<td>void</td>
<td>set_cubic_interpolation ( bool enable )</td>
</tr>
<tr>
<td>void</td>
<td>set_h_offset ( float h_offset )</td>
</tr>
<tr>
<td>void</td>
<td>set_loop ( bool loop )</td>
</tr>
<tr>
<td>void</td>
<td>set_offset ( float offset )</td>
</tr>
<tr>
<td>void</td>
<td>set_rotate ( bool enable )</td>
</tr>
<tr>
<td>void</td>
<td>set_unit_offset ( float unit_offset )</td>
</tr>
<tr>
<td>void</td>
<td>set_v_offset ( float v_offset )</td>
</tr>
</tbody>
</table>

12.205.3 Description

This node takes its parent Path2D, and returns the coordinates of a point within it, given a distance from the first vertex. It is useful for making other nodes follow a path, without coding the movement pattern. For that, the nodes must be descendants of this node. Then, when setting an offset in this node, the descendant nodes will move accordingly.

12.205.4 Member Function Description

- **bool get_cubic_interpolation ( ) const**
  This method returns whether the position between two cached points (see set_cubic_interpolation) is interpolated linearly, or cubicly.

- **float get_h_offset ( ) const**
  Returns the horizontal displacement this node has from its parent Path2D.

- **float get_offset ( ) const**
  Returns the distance along the path in pixels.

- **float get_unit_offset ( ) const**
  Returns the distance along the path as a number in the range 0.0 (for the first vertex) to 1.0 (for the last).
Returns the vertical displacement this node has from its parent Path2D.

- `bool has_loop()` const

Returns whether this node wraps its offsets around, or truncates them to the path ends.

- `bool is_rotating()` const

Returns whether this node rotates to follow the path.

- `void set_cubic_interpolation(bool enable)`

The points along the Curve2D of the Path2D are precomputed before use, for faster calculations. The point at the requested offset is then calculated interpolating between two adjacent cached points. This may present a problem if the curve makes sharp turns, as the cached points may not follow the curve closely enough.

There are two answers to this problem: Either increase the number of cached points and increase memory consumption, or make a cubic interpolation between two points at the cost of (slightly) slower calculations.

This method controls whether the position between two cached points is interpolated linearly, or cubicly.

- `void set_h_offset(float h_offset)`

Moves this node horizontally. As this node’s position will be set every time its offset is set, this allows many PathFollow2D to share the same curve (and thus the same movement pattern), yet not return the same position for a given path offset.

A similar effect may be achieved moving this node’s descendants.

- `void set_loop(bool loop)`

If set, any offset outside the path’s length (whether set by `set_offset` or `set_unit_offset` will wrap around, instead of stopping at the ends. Set it for cyclic paths.

- `void set_offset(float offset)`

Sets the distance from the first vertex, measured in pixels along the path. This sets this node’s position to a point within the path.

- `void set_rotate(bool enable)`

If set, this node rotates to follow the path, making its descendants rotate.

- `void set_unit_offset(float unit_offset)`

Sets the distance from the first vertex, considering 0.0 as the first vertex and 1.0 as the last. This is just another way of expressing the offset within the path, as the offset supplied is multiplied internally by the path’s length.

- `void set_v_offset(float v_offset)`

Moves the PathFollow2D vertically, for the same reasons of `set_h_offset`.

12.206 PathRemap

**Inherits:** Object

**Category:** Core

12.206.1 Brief Description

Singleton containing the list of remapped resources.
12.206.2 Member Functions

| void | add_remap ( String from, String to, String locale="") |
| void | clear_remaps () |
| void | erase_remap ( String path ) |
| String | get_remap ( String path ) const |
| bool | has_remap ( String path ) const |

12.206.3 Description

When exporting, the types of some resources may change internally so they are converted to more optimized versions. While it’s not usually necessary to access to this directly (path remapping happens automatically when opening a file), it’s exported just for information.

12.206.4 Member Function Description

- void add_remap ( String from, String to, String locale="")
  Add a remap from a file to another.

- void clear_remaps ()
  Clear all remaps.

- void erase_remap ( String path )
  Erase a remap.

- String get_remap ( String path ) const
  Return the remapped new path of a file.

- bool has_remap ( String path ) const
  Return true if a file is being remapped.

12.207 PCKPacker

Inherits: Reference < Object

Category: Core

12.207.1 Brief Description

12.207.2 Member Functions

| int | add_file ( String pck_path, String source_path ) |
| int | flush ( bool verbose ) |
| int | pck_start ( String pck_name, int alignment ) |
12.207.3 Member Function Description

- `int add_file (String pck_path, String source_path )`
- `int flush (bool verbose )`
- `int pck_start (String pck_name, int alignment )`

12.208 Performance

Inherits: `Object`

Category: Core

12.208.1 Brief Description

12.208.2 Member Functions

```cpp
float get_monitor (int monitor) const
```

12.208.3 Numeric Constants

- `TIME_FPS = 0`
- `TIME_PROCESS = 1`
- `TIME_FIXED_PROCESS = 2`
- `MEMORY_STATIC = 3`
- `MEMORY_DYNAMIC = 4`
- `MEMORY_STATIC_MAX = 5`
- `MEMORY_DYNAMIC_MAX = 6`
- `MEMORY_MESSAGE_BUFFER_MAX = 7`
- `OBJECT_COUNT = 8`
- `OBJECT_RESOURCE_COUNT = 9`
- `OBJECT_NODE_COUNT = 10`
- `RENDER_OBJECTS_IN_FRAME = 11`
- `RENDER_VERTICES_IN_FRAME = 12`
- `RENDER_MATERIAL_CHANGES_IN_FRAME = 13`
- `RENDER_SHADER_CHANGES_IN_FRAME = 14`
- `RENDER_SURFACE_CHANGES_IN_FRAME = 15`
- `RENDER_DRAW_CALLS_IN_FRAME = 16`
- `RENDER_USAGE_VIDEO_MEM_TOTAL = 20`
- `RENDER_VIDEO_MEM_USED = 17`
- `RENDER_TEXTURE_MEM_USED = 18`
• RENDER_VERTEX_MEM_USED = 19
• PHYSICS_2D_ACTIVE_OBJECTS = 21
• PHYSICS_2D_COLLISION_PAIRS = 22
• PHYSICS_2D_ISLAND_COUNT = 23
• PHYSICS_3D_ACTIVE_OBJECTS = 24
• PHYSICS_3D_COLLISION_PAIRS = 25
• PHYSICS_3D_ISLAND_COUNT = 26
• MONITOR_MAX = 27

12.208.4 Member Function Description

• float get_monitor ( int monitor ) const

12.209 PHashTranslation

Inherits: Translation < Resource < Reference < Object
Category: Core

12.209.1 Brief Description

Optimized translation.

12.209.2 Member Functions

void generate ( Translation from )

12.209.3 Description

Optimized translation. Uses real-time compressed translations, which results in very small dictionaries.

12.209.4 Member Function Description

• void generate ( Translation from )

12.210 Physics2DDirectBodyState

Inherits: Object
Inherited By: Physics2DDirectBodyStateSW
Category: Core
12.210.1 Brief Description

Direct access object to a physics body in the Physics2DServer.

12.210.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>float</td>
<td>get_angular_velocity() const</td>
<td>Return the angular velocity of the body.</td>
</tr>
<tr>
<td>RID</td>
<td>get_contact_collider(contact_idx) const</td>
<td>Return the RID of the collider.</td>
</tr>
<tr>
<td>int</td>
<td>get_contact_collider_id(contact_idx) const</td>
<td>Return the object id of the collider.</td>
</tr>
<tr>
<td>Object</td>
<td>get_contact_collider_object(contact_idx) const</td>
<td></td>
</tr>
<tr>
<td>Vector2</td>
<td>get_contact_collider_pos(contact_idx) const</td>
<td></td>
</tr>
<tr>
<td>int</td>
<td>get_contact_collider_shape(contact_idx) const</td>
<td></td>
</tr>
<tr>
<td>Variant</td>
<td>get_contact_collider_shape_metadata(contact_idx) const</td>
<td></td>
</tr>
<tr>
<td>Vector2</td>
<td>get_contact_collider_velocity_at_pos(contact_idx) const</td>
<td></td>
</tr>
<tr>
<td>int</td>
<td>get_contact_count() const</td>
<td></td>
</tr>
<tr>
<td>Vector2</td>
<td>get_contact_local_normal(contact_idx) const</td>
<td></td>
</tr>
<tr>
<td>Vector2</td>
<td>get_contact_local_pos(contact_idx) const</td>
<td></td>
</tr>
<tr>
<td>int</td>
<td>get_contact_local_shape(contact_idx) const</td>
<td></td>
</tr>
<tr>
<td>float</td>
<td>get_inverse_inertia() const</td>
<td></td>
</tr>
<tr>
<td>float</td>
<td>get_inverse_mass() const</td>
<td></td>
</tr>
<tr>
<td>float</td>
<td>get_linear_velocity() const</td>
<td></td>
</tr>
<tr>
<td>float</td>
<td>get_step() const</td>
<td></td>
</tr>
<tr>
<td>float</td>
<td>get_total_angular_damp() const</td>
<td></td>
</tr>
<tr>
<td>Vector2</td>
<td>get_total_gravity() const</td>
<td></td>
</tr>
<tr>
<td>float</td>
<td>get_total_linear_damp() const</td>
<td></td>
</tr>
<tr>
<td>Matrix32</td>
<td>get_transform() const</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>integrate_forces()</td>
<td></td>
</tr>
<tr>
<td>bool</td>
<td>is_sleeping() const</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>set_angular_velocity(velocity)</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>set_linear_velocity(velocity)</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>set_sleep_state(enabled)</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>set_transform(transform)</td>
<td></td>
</tr>
</tbody>
</table>

12.210.3 Description

Direct access object to a physics body in the Physics2DServer. This object is passed via the direct state callback of rigid/character bodies, and is intended for changing the direct state of that body.

12.210.4 Member Function Description

- **float get_angular_velocity() const**
  Return the angular velocity of the body.

- **RID get_contact_collider(contact_idx) const**
  Return the RID of the collider.

- **int get_contact_collider_id(contact_idx) const**
  Return the object id of the collider.
• **Object** `get_contactColliderObject (int contact_idx) const`
  Return the collider object, this depends on how it was created (will return a scene node if such was used to create it).

• **Vector2** `get_contactColliderPos (int contact_idx) const`
  Return the contact position in the collider.

• **int** `get_contactColliderShape (int contact_idx) const`
  Return the collider shape index.

• **Variant** `get_contactColliderShapeMetadata (int contact_idx) const`
  Return the metadata of the collided shape. This metadata is different from `Object.get_meta`, and is set with `Physics2DServer.shape_set_data`.

• **Vector2** `get_contactColliderVelocityAtPos (int contact_idx) const`
  Return the linear velocity vector at contact point of the collider.

• **int** `get_contactCount () const`
  Return the amount of contacts this body has with other bodies. Note that by default this returns 0 unless bodies are configured to log contacts.

• **Vector2** `get_contactLocalNormal (int contact_idx) const`
  Return the local normal (of this body) of the contact point.

• **Vector2** `get_contactLocalPos (int contact_idx) const`
  Return the local position (of this body) of the contact point.

• **int** `get_contactLocalShape (int contact_idx) const`
  Return the local shape index of the collision.

• **float** `get_inverseInertia () const`
  Return the inverse of the inertia of the body.

• **float** `get_inverseMass () const`
  Return the inverse of the mass of the body.

• **Vector2** `get_linearVelocity () const`
  Return the current linear velocity of the body.

• **Physics2DDirectSpaceState** `get_spaceState ()`
  Return the current state of space, useful for queries.

• **float** `get_step () const`
  Return the timestep (delta) used for the simulation.

• **float** `get_total_angular_damp () const`
  Return the rate at which the body stops rotating, if there are not any other forces moving it.

• **Vector2** `get_total_gravity () const`
  Return the total gravity vector being currently applied to this body.

• **float** `get_total_linear_damp () const`
  Return the rate at which the body stops moving, if there are not any other forces moving it.

• **Matrix32** `get_transform () const`
Return the transform matrix of the body.

- void integrate_forces()

Call the built-in force integration code.

- bool is_sleeping() const

Return true if this body is currently sleeping (not active).

- void set_angular_velocity(float velocity)

Change the angular velocity of the body.

- void set_linear_velocity(Vector2 velocity)

Change the linear velocity of the body.

- void set_sleep_state(bool enabled)

Set the sleeping state of the body, only affects character/rigid bodies.

- void set_transform(Matrix32 transform)

Change the transform matrix of the body.

12.211 Physics2DDirectBodyStateSW

Inherits: Physics2DDirectBodyState < Object

Category: Core

12.211.1 Brief Description

Software implementation of Physics2DDirectBodyState.

12.211.2 Description

Software implementation of Physics2DDirectBodyState. This object exposes no new methods or properties and should not be used, as Physics2DDirectBodyState selects the best implementation available.

12.212 Physics2DDirectSpaceState

Inherits: Object

Category: Core

12.212.1 Brief Description

Direct access object to a space in the Physics2DServer.
### 12.212.2 Member Functions

<table>
<thead>
<tr>
<th>Array</th>
<th>cast_motion (Physics2DShapeQueryParameters shape)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>collide_shape (Physics2DShapeQueryParameters shape, int max_results=32)</td>
</tr>
<tr>
<td>Dictionary</td>
<td>get_rest_info (Physics2DShapeQueryParameters shape)</td>
</tr>
<tr>
<td>Array</td>
<td>intersect_point (Vector2 point, int max_results=32, Array exclude=Array(), int layer_mask=2147483647, int type_mask=15)</td>
</tr>
<tr>
<td>Dictionary</td>
<td>intersect_ray (Vector2 from, Vector2 to, Array exclude=Array(), int layer_mask=2147483647, int type_mask=15)</td>
</tr>
<tr>
<td>Array</td>
<td>intersect_shape (Physics2DShapeQueryParameters shape, int max_results=32)</td>
</tr>
</tbody>
</table>

### 12.212.3 Numeric Constants

- TYPE_MASK_STATIC_BODY = 1 — Check for collisions with static bodies.
- TYPE_MASK_KINEMATIC_BODY = 2 — Check for collisions with kinematic bodies.
- TYPE_MASK_RIGID_BODY = 4 — Check for collisions with rigid bodies.
- TYPE_MASK_CHARACTER_BODY = 8 — Check for collisions with rigid bodies in character mode.
- TYPE_MASK_AREA = 16 — Check for collisions with areas.
- TYPE_MASK_COLLISION = 15 — Check for collisions with any kind of bodies (but not areas).

### 12.212.4 Description

Direct access object to a space in the Physics2DServer. It’s used mainly to do queries against objects and areas residing in a given space.

### 12.212.5 Member Function Description

- **Array cast_motion (Physics2DShapeQueryParameters shape)**
  
  Check whether the shape can travel to a point. If it can, the method will return an array with two floats: The first is the distance the shape can move in that direction without colliding, and the second is the distance at which it will collide.

  If the shape can not move, the array will be empty.

- **Array collide_shape (Physics2DShapeQueryParameters shape, int max_results=32)**

  Check the intersections of a shape, given through a Physics2DShapeQueryParameters object, against the space. The resulting array contains a list of points where the shape intersects another. Like with intersect_shape, the number of returned results can be limited to save processing time.

- **Dictionary get_rest_info (Physics2DShapeQueryParameters shape)**

  Check the intersections of a shape, given through a Physics2DShapeQueryParameters object, against the space. If it collides with more than a shape, the nearest one is selected. The returned object is a dictionary containing the following fields:

  - pointo: Place where the shapes intersect.
  - normal: Normal of the object at the point where the shapes intersect.
  - shape: Shape index within the object against which the shape intersected.
metadata: Metadata of the shape against which the shape intersected. This metadata is different from `Object.get_meta`, and is set with `Physics2DServer.shape_set_data`.

collider_id: Id of the object against which the shape intersected.

collider: Object against which the shape intersected.

rid: RID of the object against which the shape intersected.

linear_velocity: The movement vector of the object the shape intersected, if it was a body. If it was an area, it is (0,0).

If the shape did not intersect anything, then an empty dictionary (`dir.empty()==true`) is returned instead.

- `Array intersect_point ( Vector2 point, int max_results=32, Array exclude=Array(), int layer_mask=2147483647, int type_mask=15 )`

Check whether a point is inside any shape. The shapes the point is inside of are returned in an array containing dictionaries with the following fields:

- shape: Shape index within the object the point is in.
- metadata: Metadata of the shape the point is in. This metadata is different from `Object.get_meta`, and is set with `Physics2DServer.shape_set_data`.
- collider_id: Id of the object the point is in.
- collider: Object the point is inside of.
- rid: RID of the object the point is in.

Additionally, the method can take an array of objects or RIDs that are to be excluded from collisions, a bitmask representing the physics layers to check in, and another bitmask for the types of objects to check (see `TYPE_MASK_` constants).

- `Dictionary intersect_ray ( Vector2 from, Vector2 to, Array exclude=Array(), int layer_mask=2147483647, int type_mask=15 )`

Intersect a ray in a given space. The returned object is a dictionary with the following fields:

- position: Place where ray is stopped.
- normal: Normal of the object at the point where the ray was stopped.
- shape: Shape index within the object against which the ray was stopped.
- metadata: Metadata of the shape against which the ray was stopped. This metadata is different from `Object.get_meta`, and is set with `Physics2DServer.shape_set_data`.
- collider_id: Id of the object against which the ray was stopped.
- collider: Object against which the ray was stopped.
- rid: RID of the object against which the ray was stopped.

If the ray did not intersect anything, then an empty dictionary (`dir.empty()==true`) is returned instead.

Additionally, the method can take an array of objects or RIDs that are to be excluded from collisions, a bitmask representing the physics layers to check in, and another bitmask for the types of objects to check (see `TYPE_MASK_` constants).

- `Array intersect_shape ( Physics2DShapeQueryParameters shape, int max_results=32 )`

Check the intersections of a shape, given through a `Physics2DShapeQueryParameters` object, against the space. The intersected shapes are returned in an array containing dictionaries with the following fields:

- shape: Shape index within the object the shape intersected.
metadata: Metadata of the shape intersected by the shape given through the `Physics2DShapeQueryParameters`. This metadata is different from `Object.get_meta`, and is set with `Physics2DServer.shape_set_data`.

collider_id: Id of the object the shape intersected.
collider: Object the shape intersected.
rid: `RID` of the object the shape intersected.

The number of intersections can be limited with the second parameter, to reduce the processing time.

### 12.213 Physics2DServer

**Inherits:** `Object`  
**Inherited By:** `Physics2DServerSW`  
**Category:** `Core`

#### 12.213.1 Brief Description

Physics 2D Server.

#### 12.213.2 Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void area_add_shape(RID area, RID shape, Matrix32 transform=(1, 0), (0, 1), (0, 0))</code></td>
<td></td>
</tr>
<tr>
<td><code>void area_attach_object_instance_ID(RID area, int id)</code></td>
<td></td>
</tr>
<tr>
<td><code>void area_clear_shapes(RID area)</code></td>
<td></td>
</tr>
<tr>
<td><code>RID area_create()</code></td>
<td></td>
</tr>
<tr>
<td><code>int area_get_object_instance_ID(RID area) const</code></td>
<td></td>
</tr>
<tr>
<td><code>void area_get_param(RID area, int param) const</code></td>
<td></td>
</tr>
<tr>
<td><code>RID area_get_shape(RID area, int shape_idx) const</code></td>
<td></td>
</tr>
<tr>
<td><code>int area_get_shape_count(RID area) const</code></td>
<td></td>
</tr>
<tr>
<td><code>Matrix32 area_get_shape_transform(RID area, int shape_idx) const</code></td>
<td></td>
</tr>
<tr>
<td><code>RID area_get_space(RID area) const</code></td>
<td></td>
</tr>
<tr>
<td><code>int area_get_space_override_mode(RID area) const</code></td>
<td></td>
</tr>
<tr>
<td><code>Matrix32 area_get_transform(RID area) const</code></td>
<td></td>
</tr>
<tr>
<td><code>void area_remove_shape(RID area, int shape_idx)</code></td>
<td></td>
</tr>
<tr>
<td><code>void area_set_collision_mask(RID area, int mask)</code></td>
<td></td>
</tr>
<tr>
<td><code>void area_set_layer_mask(RID area, int mask)</code></td>
<td></td>
</tr>
<tr>
<td><code>void area_set_monitor_callback(RID area, Object receiver, String method)</code></td>
<td></td>
</tr>
<tr>
<td><code>void area_set_param(RID area, int param, Variant value)</code></td>
<td></td>
</tr>
<tr>
<td><code>void area_set_shape(RID area, int shape_idx, RID shape)</code></td>
<td></td>
</tr>
<tr>
<td><code>void area_set_shape_transform(RID area, int shape_idx, Matrix32 transform)</code></td>
<td></td>
</tr>
<tr>
<td><code>void area_set_space(RID area, RID space)</code></td>
<td></td>
</tr>
<tr>
<td><code>void area_set_space_override_mode(RID area, int mode)</code></td>
<td></td>
</tr>
<tr>
<td><code>void area_set_transform(RID area, Matrix32 transform)</code></td>
<td></td>
</tr>
<tr>
<td><code>void body_add_collision_exception(RID body, RID excepted_body)</code></td>
<td></td>
</tr>
<tr>
<td><code>void body_add_force(RID body, Vector2 offset, Vector2 force)</code></td>
<td></td>
</tr>
<tr>
<td><code>void body_add_shape(RID body, RID shape, Matrix32 transform=(1, 0), (0, 1), (0, 0))</code></td>
<td></td>
</tr>
<tr>
<td><code>void body_apply_impulse(RID body, Vector2 pos, Vector2 impulse)</code></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>void body_attach_object_instance_ID (RID body, int id)</code></td>
<td>Attach an object instance ID to a body.</td>
</tr>
<tr>
<td><code>void body_clear_shapes (RID body)</code></td>
<td>Clear all shapes from a body.</td>
</tr>
<tr>
<td><code>RID body_create (int mode=2, bool init_sleeving=false)</code></td>
<td>Create a new body with specified parameters.</td>
</tr>
<tr>
<td><code>int body_get_collision_mask (RID body)</code> const</td>
<td>Get the collision mask of a body.</td>
</tr>
<tr>
<td><code>int body_get_continuous_collision_detection_mode (RID body)</code> const</td>
<td>Get the continuous collision detection mode of a body.</td>
</tr>
<tr>
<td><code>int body_get_layer_mask (RID body)</code> const</td>
<td>Get the layer mask of a body.</td>
</tr>
<tr>
<td><code>int body_get_max_contacts_reported (RID body)</code> const</td>
<td>Get the maximum number of contacts reported for a body.</td>
</tr>
<tr>
<td><code>int body_get_mode (RID body)</code> const</td>
<td>Get the mode of a body.</td>
</tr>
<tr>
<td><code>Vector2 body_get_one_way_collision_direction (RID body)</code> const</td>
<td>Get the one-way collision direction of a body.</td>
</tr>
<tr>
<td><code>float body_get_one_way_collision_max_depth (RID body)</code> const</td>
<td>Get the maximum depth of a one-way collision of a body.</td>
</tr>
<tr>
<td><code>float body_get_param (RID body, int param)</code> const</td>
<td>Get a parameter of a body.</td>
</tr>
<tr>
<td><code>RID body_get_shape (RID body, int shape_idx)</code> const</td>
<td>Get a shape by index from a body.</td>
</tr>
<tr>
<td><code>int body_get_shape_count (RID body)</code> const</td>
<td>Get the number of shapes in a body.</td>
</tr>
<tr>
<td><code>void body_get_shape_metadata (RID body, int shape_idx)</code> const</td>
<td>Get the metadata of a shape in a body.</td>
</tr>
<tr>
<td><code>Matrix2 body_get_shape_transform (RID body, int shape_idx)</code> const</td>
<td>Get the transform of a shape in a body.</td>
</tr>
<tr>
<td><code>RID body_get_space (RID body)</code> const</td>
<td>Get the space of a body.</td>
</tr>
<tr>
<td><code>void body_set_state (RID body, int state)</code> const</td>
<td>Set the state of a body.</td>
</tr>
<tr>
<td><code>bool body_is_omitting_force_integration (RID body)</code> const</td>
<td>Check if a body is omitting force integration.</td>
</tr>
<tr>
<td><code>bool body_is_shape_set_as_trigger (RID body, int shape_idx)</code> const</td>
<td>Check if a shape is set as a trigger.</td>
</tr>
<tr>
<td><code>void body_remove_collision_exception (RID body, RID excepted_body)</code></td>
<td>Remove an exception collision.</td>
</tr>
<tr>
<td><code>void body_remove_shape (RID body, int shape_idx)</code></td>
<td>Remove a shape from a body.</td>
</tr>
<tr>
<td><code>void body_set_axis_velocity (RID body, Vector2 axis_velocity)</code></td>
<td>Set the axis velocity of a body.</td>
</tr>
<tr>
<td><code>void body_set_collision_mask (RID body, int mask)</code></td>
<td>Set the collision mask of a body.</td>
</tr>
<tr>
<td><code>void body_set_continuous_collision_detection_mode (RID body, int mode)</code></td>
<td>Set the continuous collision detection mode of a body.</td>
</tr>
<tr>
<td><code>void body_set_force_integration_callback (RID body, Object receiver, String method, Variant userdata=NULL)</code></td>
<td>Set a callback for force integration.</td>
</tr>
<tr>
<td><code>void body_set_layer_mask (RID body, int mask)</code></td>
<td>Set the layer mask of a body.</td>
</tr>
<tr>
<td><code>void body_set_max_contacts_reported (RID body, int amount)</code></td>
<td>Set the maximum number of contacts reported for a body.</td>
</tr>
<tr>
<td><code>void body_set_mode (RID body, int mode)</code></td>
<td>Set the mode of a body.</td>
</tr>
<tr>
<td><code>void body_set_omit_force_integration (RID body, bool enable)</code></td>
<td>Enable or disable force integration.</td>
</tr>
<tr>
<td><code>void body_set_one_way_collision_direction (RID body, Vector2 normal)</code></td>
<td>Set the one-way collision direction of a body.</td>
</tr>
<tr>
<td><code>void body_set_one_way_collision_max_depth (RID body, float depth)</code></td>
<td>Set the maximum depth of a one-way collision of a body.</td>
</tr>
<tr>
<td><code>void body_set_param (RID body, int param, float value)</code></td>
<td>Set a parameter of a body.</td>
</tr>
<tr>
<td><code>void body_set_shape (RID body, int shape_idx, RID shape)</code></td>
<td>Set a shape by index to a body.</td>
</tr>
<tr>
<td><code>void body_set_shape_as_trigger (RID body, int shape_idx, bool enable)</code></td>
<td>Set a shape as a trigger.</td>
</tr>
<tr>
<td><code>void body_set_shape_metadata (RID body, int shape_idx, Variant metadata)</code></td>
<td>Set the metadata of a shape in a body.</td>
</tr>
<tr>
<td><code>void body_set_shape_transform (RID body, int shape_idx, Matrix2 transform)</code></td>
<td>Set the transform of a shape in a body.</td>
</tr>
<tr>
<td><code>void body_set_space (RID body, RID space)</code></td>
<td>Set the space of a body.</td>
</tr>
<tr>
<td><code>bool body_test_motion (RID body, Vector2 motion, float margin=0.08, Physics2DTestMotionResult result=NULL)</code></td>
<td>Test motion of a body.</td>
</tr>
<tr>
<td><code>bool body_test_motion_from (RID body, Matrix2 from, Vector2 motion, float margin=0.08, Physics2DTestMotionResult result=NULL)</code></td>
<td>Test motion from a body.</td>
</tr>
<tr>
<td><code>RID damped_string_joint_create (Vector2 anchor_a, Vector2 anchor_b, RID body_a, RID body_b=RID())</code></td>
<td>Create a damped string joint.</td>
</tr>
<tr>
<td><code>float damped_string_joint_get_param (RID joint, int param)</code> const</td>
<td>Get a parameter of a damped string joint.</td>
</tr>
<tr>
<td><code>void damped_string_joint_set_param (RID joint, int param, float value)</code></td>
<td>Set a parameter of a damped string joint.</td>
</tr>
<tr>
<td><code>void free_rid (RID rid)</code></td>
<td>Free a RID.</td>
</tr>
<tr>
<td><code>int get_process_info (int process_info)</code></td>
<td>Get process information.</td>
</tr>
<tr>
<td><code>RID groove_joint_create (Vector2 groove1_a, Vector2 groove2_a, Vector2 anchor_b, RID body_a=RID(), RID body_b=RID())</code></td>
<td>Create a groove joint.</td>
</tr>
<tr>
<td><code>float joint_get_param (RID joint, int param)</code> const</td>
<td>Get a parameter of a joint.</td>
</tr>
<tr>
<td><code>int joint_get_type (RID joint)</code> const</td>
<td>Get the type of a joint.</td>
</tr>
<tr>
<td><code>void joint_set_param (RID joint, int param, float value)</code></td>
<td>Set a parameter of a joint.</td>
</tr>
</tbody>
</table>
Table 12.26 – continued from previous page

<table>
<thead>
<tr>
<th>RID</th>
<th>pin_joint_create ( Vector2 anchor, RID body_a, RID body_b=RID() )</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>set_active ( bool active )</td>
</tr>
<tr>
<td>RID</td>
<td>shape_create ( int type )</td>
</tr>
<tr>
<td>void</td>
<td>shape_get_data ( RID shape ) const</td>
</tr>
<tr>
<td>int</td>
<td>shape_get_type ( RID shape ) const</td>
</tr>
<tr>
<td>void</td>
<td>shape_set_data ( RID shape, Variant data )</td>
</tr>
<tr>
<td>RID</td>
<td>space_create ( )</td>
</tr>
<tr>
<td>Physics2DDirectSpaceState</td>
<td>space_get_direct_state ( RID space )</td>
</tr>
<tr>
<td>float</td>
<td>space_get_param ( RID space, int param ) const</td>
</tr>
<tr>
<td>bool</td>
<td>space_is_active ( RID space ) const</td>
</tr>
<tr>
<td>void</td>
<td>space_set_active ( RID space, bool active )</td>
</tr>
<tr>
<td>void</td>
<td>space_set_param ( RID space, int param, float value )</td>
</tr>
</tbody>
</table>

### 12.213.3 Numeric Constants

- **SPACE_PARAM_CONTACT_RECYCLE_RADIUS = 0** — Constant to set/get the maximum distance a pair of bodies has to move before their collision status has to be recalculated.

- **SPACE_PARAM_CONTACT_MAX_SEPARATION = 1** — Constant to set/get the maximum distance a shape can be from another before they are considered separated.

- **SPACE_PARAM_BODY_MAX_ALLOWED_PENETRATION = 2** — Constant to set/get the maximum distance a shape can penetrate another shape before it is considered a collision.

- **SPACE_PARAM_BODY_LINEAR_VELOCITY_SLEEP_TRESHOLD = 3** — Constant to set/get the linear velocity threshold. Bodies slower than this will be marked as potentially inactive.

- **SPACE_PARAM_BODY_ANGULAR_VELOCITY_SLEEP_TRESHOLD = 4** — Constant to set/get the angular velocity threshold. Bodies slower than this will be marked as potentially inactive.

- **SPACE_PARAM_BODY_TIME_TO_SLEEP = 5** — Constant to set/get the maximum time of activity. A body marked as potentially inactive for both linear and angular velocity will be put to sleep after this time.

- **SPACE_PARAM_CONSTRAINT_DEFAULT_BIAS = 6** — Constant to set/get the default solver bias for all physics constraints. A solver bias is a factor controlling how much two objects “rebound”, after violating a constraint, to avoid leaving them in that state because of numerical imprecision.

- **SHAPE_LINE = 0** — This is the constant for creating line shapes. A line shape is an infinite line with an origin point, and a normal. Thus, it can be used for front/behind checks.

- **SHAPE_SEGMENT = 2** — This is the constant for creating segment shapes. A segment shape is a line from a point A to a point B. It can be checked for intersections.

- **SHAPE_CIRCLE = 3** — This is the constant for creating circle shapes. A circle shape only has a radius. It can be used for intersections and inside/outside checks.

- **SHAPE_RECTANGLE = 4** — This is the constant for creating rectangle shapes. A rectangle shape is defined by a width and a height. It can be used for intersections and inside/outside checks.

- **SHAPE_CAPSULE = 5** — This is the constant for creating capsule shapes. A capsule shape is defined by a radius and a length. It can be used for intersections and inside/outside checks.

- **SHAPE_CONVEX_POLYGON = 6** — This is the constant for creating convex polygon shapes. A polygon is defined by a list of points. It can be used for intersections and inside/outside checks. Unlike the method `CollisionPolygon2D.set_polygon`, polygons modified with `shape_set_data` do not verify that the points supplied form, in fact, a convex polygon.
• **SHAPE_CONCAVE_POLYGON** = 7 — This is the constant for creating concave polygon shapes. A polygon is defined by a list of points. It can be used for intersections checks, but not for inside/outside checks.

• **SHAPE_CUSTOM** = 8 — This constant is used internally by the engine. Any attempt to create this kind of shape results in an error.

• **AREA_PARAM_GRAVITY** = 0 — Constant to set/get gravity strength in an area.

• **AREA_PARAM_GRAVITY_VECTOR** = 1 — Constant to set/get gravity vector/center in an area.

• **AREA_PARAM_GRAVITY_IS_POINT** = 2 — Constant to set/get whether the gravity vector of an area is a direction, or a center point.

• **AREA_PARAM_GRAVITY_DISTANCE_SCALE** = 3 — Constant to set/get the falloff factor for point gravity of an area. The greater this value is, the faster the strength of gravity decreases with the square of distance.

• **AREA_PARAM_GRAVITY_POINT_ATTENUATION** = 4 — This constant was used to set/get the falloff factor for point gravity. It has been superseded by **AREA_PARAM_GRAVITY_DISTANCE_SCALE**.

• **AREA_PARAM_LINEAR_DAMP** = 5 — Constant to set/get the linear dampening factor of an area.

• **AREA_PARAM_ANGULAR_DAMP** = 6 — Constant to set/get the angular dampening factor of an area.

• **AREA_PARAM_PRIORITY** = 7 — Constant to set/get the priority (order of processing) of an area.

• **AREA_SPACE_OVERRIDE_DISABLED** = 0 — This area does not affect gravity/damp. These are generally areas that exist only to detect collisions, and objects entering or exiting them.

• **AREA_SPACE_OVERRIDE_COMBINE** = 1 — This area adds its gravity/damp values to whatever has been calculated so far. This way, many overlapping areas can combine their physics to make interesting effects.

• **AREA_SPACE_OVERRIDE_COMBINE_REPLACE** = 2 — This area adds its gravity/damp values to whatever has been calculated so far. Then stops taking into account the rest of the areas, even the default one.

• **AREA_SPACE_OVERRIDE_REPLACE** = 3 — This area replaces any gravity/damp, even the default one, and stops taking into account the rest of the areas.

• **AREA_SPACE_OVERRIDE_REPLACE_COMBINE** = 4 — This area replaces any gravity/damp calculated so far, but keeps calculating the rest of the areas, down to the default one.

• **BODY_MODE_STATIC** = 0 — Constant for static bodies.

• **BODY_MODE_KINEMATIC** = 1 — Constant for kinematic bodies.

• **BODY_MODE_RIGID** = 2 — Constant for rigid bodies.

• **BODY_MODE_CHARACTER** = 3 — Constant for rigid bodies in character mode. In this mode, a body can not rotate, and only its linear velocity is affected by physics.

• **BODY_PARAM_BOUNCE** = 0 — Constant to set/get a body’s bounce factor.

• **BODY_PARAM_FRICTION** = 1 — Constant to set/get a body’s friction.

• **BODY_PARAM_MASS** = 2 — Constant to set/get a body’s mass.

• **BODY_PARAM_INERTIA** = 3 — Constant to set/get a body’s inertia.

• **BODY_PARAM_GRAVITY_SCALE** = 4 — Constant to set/get a body’s gravity multiplier.

• **BODY_PARAM_LINEAR_DAMP** = 5 — Constant to set/get a body’s linear dampening factor.

• **BODY_PARAM_ANGULAR_DAMP** = 6 — Constant to set/get a body’s angular dampening factor.

• **BODY_PARAM_MAX** = 7 — This is the last ID for body parameters. Any attempt to set this property is ignored. Any attempt to get it returns 0.

• **BODY_STATE_TRANSFORM** = 0 — Constant to set/get the current transform matrix of the body.
• **BODY_STATE_LINEAR_VELOCITY** = 1 — Constant to set/get the current linear velocity of the body.
• **BODY_STATE_ANGULAR_VELOCITY** = 2 — Constant to set/get the current angular velocity of the body.
• **BODY_STATE_SLEEPING** = 3 — Constant to sleep/wake up a body, or to get whether it is sleeping.
• **BODY_STATE_CAN_SLEEP** = 4 — Constant to set/get whether the body can sleep.
• **JOINT_PIN** = 0 — Constant to create pin joints.
• **JOINT_GROOVE** = 1 — Constant to create groove joints.
• **JOINT_DAMPED_SPRING** = 2 — Constant to create damped spring joints.
• **DAMPED_STRING_REST_LENGTH** = 0 — Set the resting length of the spring joint. The joint will always try to go to back this length when pulled apart.
• **DAMPED_STRING_STIFFNESS** = 1 — Set the stiffness of the spring joint. The joint applies a force equal to the stiffness times the distance from its resting length.
• **DAMPED_STRING_DAMPING** = 2 — Set the damping ratio of the spring joint. A value of 0 indicates an undamped spring, while 1 causes the system to reach equilibrium as fast as possible (critical damping).
• **CCD_MODE_DISABLED** = 0 — Disables continuous collision detection. This is the fastest way to detect body collisions, but can miss small, fast-moving objects.
• **CCD_MODE_CAST_RAY** = 1 — Enables continuous collision detection by raycasting. It is faster than shapecasting, but less precise.
• **CCD_MODE_CAST_SHAPE** = 2 — Enables continuous collision detection by shapecasting. It is the slowest CCD method, and the most precise.
• **AREA_BODY_ADDED** = 0 — The value of the first parameter and area callback function receives, when an object enters one of its shapes.
• **AREA_BODY_REMOVED** = 1 — The value of the first parameter and area callback function receives, when an object exits one of its shapes.
• **INFO_ACTIVE_OBJECTS** = 0 — Constant to get the number of objects that are not sleeping.
• **INFO_COLLISION_PAIRS** = 1 — Constant to get the number of possible collisions.
• **INFO_ISLAND_COUNT** = 2 — Constant to get the number of space regions where a collision could occur.

### 12.213.4 Description

Physics 2D Server is the server responsible for all 2D physics. It can create many kinds of physics objects, but does not insert them on the node tree.

### 12.213.5 Member Function Description

- void **area_add_shape** ( RID area, RID shape, Matrix32 transform=((1, 0), (0, 1), (0, 0)) )
  Add a shape to the area, along with a transform matrix. Shapes are usually referenced by their index, so you should track which shape has a given index.

- void **area_attach_object_instance_ID** ( RID area, int id )
  Assign the area to a descendant of Object, so it can exist in the node tree.

- void **area_clear_shapes** ( RID area )
  Remove all shapes from an area. It does not delete the shapes, so they can be reassigned later.
• **RID area_create()**

Create an Area2D.

• **int area_get_object_instance_ID(RID area) const**

Get the instance ID of the object the area is assigned to.

• **void area_get_param(RID area, int param) const**

Return an area parameter value.

• **RID area_get_shape(RID area, int shape_idx) const**

Return the RID of the nth shape of an area.

• **int area_get_shape_count(RID area) const**

Return the number of shapes assigned to an area.

• **Matrix32 area_get_shape_transform(RID area, int shape_idx) const**

Return the transform matrix of a shape within an area.

• **RID area_get_space(RID area) const**

Return the space assigned to the area.

• **int area_get_space_override_mode(RID area) const**

Return the space override mode for the area.

• **Matrix32 area_get_transform(RID area) const**

Return the transform matrix for an area.

• **void area_remove_shape(RID area, int shape_idx)**

Remove a shape from an area. It does not delete the shape, so it can be reassigned later.

• **void area_set_collision_mask(RID area, int mask)**

Set which physics layers the area will monitor.

• **void area_set_layer_mask(RID area, int mask)**

Assign the area to one or many physics layers.

• **void area_set_monitor_callback(RID area, Object receiver, String method)**

Set the function to call when any body/area enters or exits the area. This callback will be called for any object interacting with the area, and takes five parameters:

1: AREA_BODY_ADDED or AREA_BODY_REMOVED, depending on whether the object entered or exited the area.

2: RID of the object that entered/exited the area.

3: Instance ID of the object that entered/exited the area.

4: The shape index of the object that entered/exited the area.

5: The shape index of the area where the object entered/exited.

• **void area_set_param(RID area, int param, Variant value)**

Set the value for an area parameter. A list of available parameters is on the AREA_PARAM_* constants.

• **void area_set_shape(RID area, int shape_idx, RID shape)**

Substitute a given area shape by another. The old shape is selected by its index, the new one by its RID.
• void `area_set_shape_transform` ( `RID` area, `int` shape_idx, `Matrix32` transform )
Set the transform matrix for an area shape.

• void `area_set_space` ( `RID` area, `RID` space )
Assign a space to the area.

• void `area_set_space_override_mode` ( `RID` area, `int` mode )
Set the space override mode for the area. The modes are described in the constants `AREA_SPACE_OVERRIDE_*`.

• void `area_set_transform` ( `RID` area, `Matrix32` transform )
Set the transform matrix for an area.

• void `body_add_collision_exception` ( `RID` body, `RID` excepted_body )
Add a body to the list of bodies exempt from collisions.

• void `body_add_force` ( `RID` body, `Vector2` offset, `Vector2` force )
Add a positioned force to the applied force and torque. As with `body_apply_impulse`, both the force and the offset from the body origin are in global coordinates. A force differs from an impulse in that, while the two are forces, the impulse clears itself after being applied.

• void `body_add_shape` ( `RID` body, `RID` shape, `Matrix32` transform=((1, 0), (0, 1), (0, 0)) )
Add a shape to the body, along with a transform matrix. Shapes are usually referenced by their index, so you should track which shape has a given index.

• void `body_apply_impulse` ( `RID` body, `Vector2` pos, `Vector2` impulse )
Add a positioned impulse to the applied force and torque. Both the force and the offset from the body origin are in global coordinates.

• void `body_attach_object_instance_ID` ( `RID` body, `int` id )
Assign the area to a descendant of `Object`, so it can exist in the node tree.

• void `body_clear_shapes` ( `RID` body )
Remove all shapes from a body.

• `RID` `body_create` ( `int` mode=2, `bool` init_sleeping=false )
Create a physics body. The first parameter can be any value from constants `BODY_MODE*`, for the type of body created. Additionally, the body can be created in sleeping state to save processing time.

• `int` `body_get_collision_mask` ( `RID` body ) const
Return the physics layer or layers a body can collide with.

• `int` `body_get_continuous_collision_detection_mode` ( `RID` body ) const
Return the continuous collision detection mode.

• `int` `body_get_layer_mask` ( `RID` body ) const
Return the physics layer or layers a body belongs to.

• `int` `body_get_max_contacts_reported` ( `RID` body ) const
Return the maximum contacts that can be reported. See `body_set_max_contacts_reported`.

• `int` `body_get_mode` ( `RID` body ) const
Return the body mode.

• `int` `body_get_object_instance_ID` ( `RID` body ) const
Get the instance ID of the object the area is assigned to.

- **Vector2** `body_get_one_way_collision_direction ( RID body ) const`
  Return the direction used for one-way collision detection.

- **float** `body_get_one_way_collision_max_depth ( RID body ) const`
  Return how far a body can go through the given one, when it allows one-way collisions.

- **float** `body_get_param ( RID body, int param ) const`
  Return the value of a body parameter.

- **RID** `body_get_shape ( RID body, int shape_idx ) const`
  Return the **RID** of the nth shape of a body.

- **int** `body_get_shape_count ( RID body ) const`
  Return the number of shapes assigned to a body.

- **void** `body_get_shape_metadata ( RID body, int shape_idx ) const`
  Return the metadata of a shape of a body.

- **Matrix32** `body_get_shape_transform ( RID body, int shape_idx ) const`
  Return the transform matrix of a body shape.

- **RID** `body_get_space ( RID body ) const`
  Return the **RID** of the space assigned to a body.

- **void** `body_get_state ( RID body, int state ) const`
  Return a body state.

- **bool** `body_is_omitting_force_integration ( RID body ) const`
  Return whether a body uses a callback function to calculate its own physics (see `body_set_force_integration_callback`).

- **bool** `body_is_shape_set_as_trigger ( RID body, int shape_idx ) const`
  Return whether a body’s shape is marked as a trigger.

- **void** `body_remove_collision_exception ( RID body, RID excepted_body )`
  Remove a body from the list of bodies exempt from collisions.

- **void** `body_remove_shape ( RID body, int shape_idx )`
  Remove a shape from a body. The shape is not deleted, so it can be reused afterwards.

- **void** `body_set_axis_velocity ( RID body, Vector2 axis_velocity )`
  Set an axis velocity. The velocity in the given vector axis will be set as the given vector length. This is useful for jumping behavior.

- **void** `body_set_collision_mask ( RID body, int mask )`
  Set the physics layer or layers a body can collide with.

- **void** `body_set_continuous_collision_detection_mode ( RID body, int mode )`
  Set the continuous collision detection mode from any of the CCD_MODE_* constants.

Continuous collision detection tries to predict where a moving body will collide, instead of moving it and correcting its movement if it collided.
• void **body_set_force_integration_callback** ( RID body, Object receiver, String method, Variant user-data=NULL )

Set the function used to calculate physics for an object, if that object allows it (see body_set_omit_force_integration).

• void **body_set_layer_mask** ( RID body, int mask )

Set the physics layer or layers a body belongs to.

• void **body_set_max_contacts_reported** ( RID body, int amount )

Set the maximum contacts to report. Bodies can keep a log of the contacts with other bodies, this is enabled by setting the maximum amount of contacts reported to a number greater than 0.

• void **body_set_mode** ( RID body, int mode )

Set the body mode, from one of the constants BODY_MODE*.

• void **body_set_omit_force_integration** ( RID body, bool enable )

Set whether a body uses a callback function to calculate its own physics (see body_set_force_integration_callback).

• void **body_set_one_way_collision_direction** ( RID body, Vector2 normal )

Set a direction from which bodies can go through the given one; that is, the passed vector is the normal of the pass-through side of the surface. If this value is different from (0,0), any movement within 90 degrees of the opposite of this vector is considered an valid movement. Set this direction to (0,0) to disable one-way collisions.

• void **body_set_one_way_collision_max_depth** ( RID body, float depth )

Set how deep at most a body can be with respect to the given one for the physics server to force it to a non-overlapping position, if it allows one-way collisions (see body_set_one_way_collision_direction).

• void **body_set_param** ( RID body, int param, float value )

Set a body parameter (see BODY_PARAM* constants).

• void **body_set_shape** ( RID body, int shape_idx, RID shape )

Substitute a given body shape by another. The old shape is selected by its index, the new one by its RID.

• void **body_set_shape_as_trigger** ( RID body, int shape_idx, bool enable )

Mark a body’s shape as a trigger. A trigger shape cannot affect other bodies, but detects other shapes entering and exiting it.

• void **body_set_shape_metadata** ( RID body, int shape_idx, Variant metadata )

Set metadata of a shape within a body. This metadata is different from Object.set_meta, and can be retrieved on shape queries.

• void **body_set_shape_transform** ( RID body, int shape_idx, Matrix32 transform )

Set the transform matrix for a body shape.

• void **body_set_space** ( RID body, RID space )

Assign a space to the body (see create_space).

• void **body_set_state** ( RID body, int state, Variant value )

Set a body state (see BODY_STATE* constants).

• bool **body_test_motion** ( RID body, Vector2 motion, float margin=0.08, Physics2DTestMotionResult result=NULL )

Return whether a body can move in a given direction. Apart from the boolean return value, a Physics2DTestMotionResult can be passed to return additional information in.
• `bool body_test_motion_from ( RID body, Matrix32 from, Vector2 motion, float margin=0.08, Physics2DTestMotionResult result=NULL )`

Create a damped spring joint between two bodies. If not specified, the second body is assumed to be the joint itself.

• `float damped_string_joint_get_param ( RID joint, int param ) const`

Return the value of a damped spring joint parameter.

• `void damped_string_joint_set_param ( RID joint, int param, float value )`

Set a damped spring joint parameter. Parameters are explained in the DAMPED_STRING* constants.

• `void free_rid ( RID rid )`

Destroy any of the objects created by Physics2DServer. If the RID passed is not one of the objects that can be created by Physics2DServer, an error will be sent to the console.

• `int get_process_info ( int process_info )`

Return information about the current state of the 2D physics engine. The states are listed under the INFO_* constants.

• `RID groove_joint_create ( Vector2 groove1_a, Vector2 groove2_a, Vector2 anchor_b, RID body_a=RID(), RID body_b=RID() )`

Create a groove joint between two bodies. If not specified, the bodyies are assumed to be the joint itself.

• `float joint_get_param ( RID joint, int param ) const`

Return the value of a joint parameter.

• `int joint_get_type ( RID joint ) const`

Return the type of a joint (see JOINT_* constants).

• `void joint_set_param ( RID joint, int param, float value )`

Set a joint parameter. Parameters are explained in the JOINT_PARAM* constants.

• `RID pin_joint_create ( Vector2 anchor, RID body_a, RID body_b=RID() )`

Create a pin joint between two bodies. If not specified, the second body is assumed to be the joint itself.

• `void set_active ( bool active )`

Activate or deactivate the 2D physics engine.

• `RID shape_create ( int type )`

Create a shape of type SHAPE_*_. Does not assign it to a body or an area. To do so, you must use `area_set_shape` or `body_set_shape`.

• `void shape_get_data ( RID shape ) const`

Return the shape data.

• `int shape_get_type ( RID shape ) const`

Return the type of shape (see SHAPE_* constants).

• `void shape_set_data ( RID shape, Variant data )`

Set the shape data that defines its shape and size. The data to be passed depends on the kind of shape created `shape_get_type`.

• `RID space_create ( )`
Create a space. A space is a collection of parameters for the physics engine that can be assigned to an area or a body. It can be assigned to an area with \texttt{area\_set\_space}, or to a body with \texttt{body\_set\_space}.

- \texttt{Physics2DDirectSpaceState space\_get\_direct\_state ( RID space )}

Return the state of a space, a \texttt{Physics2DDirectSpaceState}. This object can be used to make collision/intersection queries.

- \texttt{float space\_get\_param ( RID space, int param ) const}

Return the value of a space parameter.

- \texttt{bool space\_is\_active ( RID space ) const}

Return whether the space is active.

- \texttt{void space\_set\_active ( RID space, bool active )}

Mark a space as active. It will not have an effect, unless it is assigned to an area or body.

- \texttt{void space\_set\_param ( RID space, int param, float value )}

Set the value for a space parameter. A list of available parameters is on the SPACE\_PARAM\_* constants.

### 12.214 Physics2DServerSW

**Inherits:** \texttt{Physics2DServer < Object}

**Category:** Core

#### 12.214.1 Brief Description

Software implementation of \texttt{Physics2DServer}.

#### 12.214.2 Description

Software implementation of \texttt{Physics2DServer}. This class exposes no new methods or properties and should not be used, as \texttt{Physics2DServer} automatically selects the best implementation available.

### 12.215 Physics2DShapeQueryParameters

**Inherits:** \texttt{Reference < Object}

**Category:** Core

#### 12.215.1 Brief Description

Parameters to be sent to a 2D shape physics query.
12.215.2 Member Functions

<table>
<thead>
<tr>
<th>Array</th>
<th>get_exclude() const</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>get_layer_mask() const</td>
</tr>
<tr>
<td>float</td>
<td>get_margin() const</td>
</tr>
<tr>
<td>Vector2</td>
<td>get_motion() const</td>
</tr>
<tr>
<td>int</td>
<td>get_object_type_mask() const</td>
</tr>
<tr>
<td>RID</td>
<td>get_shape_rid() const</td>
</tr>
<tr>
<td>Matrix32</td>
<td>get_transform() const</td>
</tr>
<tr>
<td>void</td>
<td>set_exclude(Array exclude)</td>
</tr>
<tr>
<td>void</td>
<td>set_layer_mask(int layer_mask)</td>
</tr>
<tr>
<td>void</td>
<td>set_margin(float margin)</td>
</tr>
<tr>
<td>void</td>
<td>set_motion(Vector2 motion)</td>
</tr>
<tr>
<td>void</td>
<td>set_object_type_mask(int object_type_mask)</td>
</tr>
<tr>
<td>void</td>
<td>set_shape(Shape2D shape)</td>
</tr>
<tr>
<td>void</td>
<td>set_shape_rid(RID shape)</td>
</tr>
<tr>
<td>void</td>
<td>set_transform(Matrix32 transform)</td>
</tr>
</tbody>
</table>

12.215.3 Description

This class contains the shape and other parameters for intersection/collision queries.

12.215.4 Member Function Description

- **Array get_exclude() const**
  
  Return the list of objects, or object **RIDs**, that will be excluded from collisions.

- **int get_layer_mask() const**
  
  Return the physics layer(s) the shape belongs to.

- **float get_margin() const**
  
  Return the collision margin for the shape.

- **Vector2 get_motion() const**
  
  Return the current movement speed of the shape.

- **int get_object_type_mask() const**
  
  Return the type of object the shape belongs to.

- **RID get_shape_rid() const**
  
  Return the **RID** of the shape queried.

- **Matrix32 get_transform() const**
  
  Return the transform matrix of the shape queried.

- **void set_exclude(Array exclude)**
  
  Set the list of objects, or object **RIDs**, that will be excluded from collisions.

- **void set_layer_mask(int layer_mask)**
  
  Set the physics layer(s) the shape belongs to.
• void set_margin ( float margin )
Set the collision margin for the shape. A collision margin is an amount (in pixels) that the shape will grow when computing collisions, to account for numerical imprecision.

• void set_motion ( Vector2 motion )
Set the current movement speed of the shape.

• void set_object_type_mask ( int object_type_mask )
Set the type of object the shape belongs to (see Physics2DDirectSpaceState.TYPE_MASK_*).

• void set_shape ( Shape2D shape )
Set the Shape2D that will be used for collision/intersection queries.

• void set_shape_rid ( RID shape )
Set the RID of the shape to be used in queries.

• void set_transform ( Matrix32 transform )
Set the transformation matrix of the shape. This is necessary to set its position/rotation/scale.

12.216 Physics2DShapeQueryResult

Inherits: Reference < Object

Category: Core

12.216.1 Brief Description

12.216.2 Member Functions

<table>
<thead>
<tr>
<th></th>
<th>function</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>get_result_count ( ) const</td>
</tr>
<tr>
<td>Object</td>
<td>get_result_object ( int idx ) const</td>
</tr>
<tr>
<td>int</td>
<td>get_result_object_id ( int idx ) const</td>
</tr>
<tr>
<td>int</td>
<td>get_result_object_shape ( int idx ) const</td>
</tr>
<tr>
<td>RID</td>
<td>get_result_rid ( int idx ) const</td>
</tr>
</tbody>
</table>

12.216.3 Member Function Description

• int get_result_count ( ) const
• Object get_result_object ( int idx ) const
• int get_result_object_id ( int idx ) const
• int get_result_object_shape ( int idx ) const
• RID get_result_rid ( int idx ) const
12.217 Physics2DTestMotionResult

Inherits: Reference < Object
Category: Core

12.217.1 Brief Description

12.217.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>get_collider ( ) const</td>
</tr>
<tr>
<td>int</td>
<td>get_collider_id ( ) const</td>
</tr>
<tr>
<td>RID</td>
<td>get_collider_rid ( ) const</td>
</tr>
<tr>
<td>int</td>
<td>get_collider_shape ( ) const</td>
</tr>
<tr>
<td>Vector2</td>
<td>get_collider_velocity ( ) const</td>
</tr>
<tr>
<td>Vector2</td>
<td>get_collision_normal ( ) const</td>
</tr>
<tr>
<td>Vector2</td>
<td>get_collision_point ( ) const</td>
</tr>
<tr>
<td>Vector2</td>
<td>get_motion ( ) const</td>
</tr>
<tr>
<td>Vector2</td>
<td>get_motion_remainder ( ) const</td>
</tr>
</tbody>
</table>

12.217.3 Member Function Description

- Object get_collider ( ) const
- int get_collider_id ( ) const
- RID get_collider_rid ( ) const
- int get_collider_shape ( ) const
- Vector2 get_collider_velocity ( ) const
- Vector2 get_collision_normal ( ) const
- Vector2 get_collision_point ( ) const
- Vector2 get_motion ( ) const
- Vector2 get_motion_remainder ( ) const

12.218 PhysicsBody

Inherits: CollisionObject < Spatial < Node < Object
Inherited By: VehicleBody, KinematicBody, StaticBody, RigidBody
Category: Core

12.218.1 Brief Description

Base class for different types of Physics bodies.
12.218.2 Member Functions

| void | add_collision_exception_with ( PhysicsBody body ) |
| int | get_collision_mask ( ) const |
| bool | get_collision_mask_bit ( int bit ) const |
| int | get_layer_mask ( ) const |
| bool | get_layer_mask_bit ( int bit ) const |
| void | remove_collision_exception_with ( PhysicsBody body ) |
| void | set_collision_mask ( int mask ) |
| void | set_collision_mask_bit ( int bit, bool value ) |
| void | set_layer_mask ( int mask ) |
| void | set_layer_mask_bit ( int bit, bool value ) |

12.218.3 Description

PhysicsBody is an abstract base class for implementing a physics body. All PhysicsBody types inherit from it.

12.218.4 Member Function Description

- void add_collision_exception_with ( PhysicsBody body )
- int get_collision_mask ( ) const
- bool get_collision_mask_bit ( int bit ) const
- int get_layer_mask ( ) const
- bool get_layer_mask_bit ( int bit ) const
- void remove_collision_exception_with ( PhysicsBody body )
- void set_collision_mask ( int mask )
- void set_collision_mask_bit ( int bit, bool value )
- void set_layer_mask ( int mask )
- void set_layer_mask_bit ( int bit, bool value )

12.219 PhysicsBody2D

Inherits: CollisionObject2D < Node2D < CanvasItem < Node < Object

Inherited By: RigidBody2D, StaticBody2D, KinematicBody2D

Category: Core

12.219.1 Brief Description

Base class for all objects affected by physics.
### 12.219.2 Member Functions

<table>
<thead>
<tr>
<th>void</th>
<th>add_collision_exception_with</th>
<th>(PhysicsBody2D body)</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>get_collision_mask</td>
<td>() const</td>
</tr>
<tr>
<td>bool</td>
<td>get_collision_mask_bit</td>
<td>(int bit) const</td>
</tr>
<tr>
<td>int</td>
<td>get_layer_mask</td>
<td>() const</td>
</tr>
<tr>
<td>bool</td>
<td>get_layer_mask_bit</td>
<td>(int bit) const</td>
</tr>
<tr>
<td>Vector2</td>
<td>get_one_way_collision_direction</td>
<td>() const</td>
</tr>
<tr>
<td>float</td>
<td>get_one_way_collision_max_depth</td>
<td>() const</td>
</tr>
<tr>
<td>void</td>
<td>remove_collision_exception_with</td>
<td>(PhysicsBody2D body)</td>
</tr>
<tr>
<td>void</td>
<td>set_collision_mask</td>
<td>(int mask)</td>
</tr>
<tr>
<td>void</td>
<td>set_collision_mask_bit</td>
<td>(int bit, bool value)</td>
</tr>
<tr>
<td>void</td>
<td>set_layer_mask</td>
<td>(int mask)</td>
</tr>
<tr>
<td>void</td>
<td>set_layer_mask_bit</td>
<td>(int bit, bool value)</td>
</tr>
<tr>
<td>void</td>
<td>set_one_way_collision_direction</td>
<td>(Vector2 dir)</td>
</tr>
<tr>
<td>void</td>
<td>set_one_way_collision_max_depth</td>
<td>(float depth)</td>
</tr>
</tbody>
</table>

### 12.219.3 Description

PhysicsBody2D is an abstract base class for implementing a physics body. All *Body2D types inherit from it.

### 12.219.4 Member Function Description

- **void add_collision_exception_with**(PhysicsBody2D body)
  Adds a body to the collision exception list. This list contains bodies that this body will not collide with.
  - **int get_collision_mask**( ) const
    Return the physics layers this area can scan for collisions.
  - **bool get_collision_mask_bit**(int bit) const
    Return an individual bit on the collision mask.
  - **int get_layer_mask**( ) const
    Return the physics layer this area is in.
  - **bool get_layer_mask_bit**(int bit) const
    Return an individual bit on the collision mask.
  - **Vector2 get_one_way_collision_direction**( ) const
    Return the direction used for one-way collision detection.
  - **float get_one_way_collision_max_depth**( ) const
    Return how far a body can go through this one, when it allows one-way collisions.
  - **void remove_collision_exception_with**(PhysicsBody2D body)
    Removes a body from the collision exception list.
  - **void set_collision_mask**(int mask)
    Set the physics layers this area can scan for collisions.
  - **void set_collision_mask_bit**(int bit, bool value)
Set/clear individual bits on the collision mask. This makes selecting the areas scanned easier.

- **void set_layer_mask ( int mask )**

Set the physics layers this area is in.

Collidable objects can exist in any of 32 different layers. These layers are not visual, but more of a tagging system instead. A collidable can use these layers/tags to select with which objects it can collide, using `set_collision_mask`.

A contact is detected if object A is in any of the layers that object B scans, or object B is in any layer scanned by object A.

- **void set_layer_mask_bit ( int bit, bool value )**

Set/clear individual bits on the layer mask. This makes getting a body in/out of only one layer easier.

- **void set_one_way_collision_direction ( Vector2 dir )**

Set a direction from which bodies can go through this one; that is, the passed vector is the normal of the pass-through side of the surface. If this value is different from (0,0), any movement within 90 degrees of the opposite of this vector is considered an valid movement. Set this direction to (0,0) to disable one-way collisions.

- **void set_one_way_collision_max_depth ( float depth )**

Set how deep at most a body can be with respect to this one for the physics server to force it to a non-overlapping position, if it allows one-way collisions (see `body_set_one_way_collision_direction`).

### 12.220 PhysicsDirectBodyState

**Inherits:** *Object*

**Inherited By:** *PhysicsDirectBodyStateSW*

**Category:** Core
12.220.1 Brief Description

12.220.2 Member Functions

| void       | add_force ( Vector3 force, Vector3 pos ) |
| void       | apply_impulse ( Vector3 pos, Vector3 j ) |
| Vector3    | get_angular_velocity ( ) const          |
| RID        | get_contact_collider ( int contact_idx ) const |
| int        | get_contact_collider_id ( int contact_idx ) const |
| Object     | get_contact_collider_object ( int contact_idx ) const |
| Vector3    | get_contact_collider_pos ( int contact_idx ) const |
| int        | get_contact_collider_shape ( int contact_idx ) const |
| Vector3    | get_contact_collider_velocity_at_pos ( int contact_idx ) const |
| int        | get_contact_count ( ) const              |
| Vector3    | get_contact_local_normal ( int contact_idx ) const |
| Vector3    | get_contact_local_pos ( int contact_id ) const |
| int        | get_contact_local_shape ( int contact_id ) const |
| float      | get_inverse_mass ( ) const               |
| Vector3    | get_inverse_inertia ( ) const            |
| float      | get_inverse_inertia ( ) const            |
| float      | get_total_angular_damp ( ) const         |
| Vector3    | get_total_gravity ( ) const              |
| float      | get_total_linear_damp ( ) const          |
| Transform  | get_transform ( ) const                  |
| void       | integrate_forces ( )                    |
| bool       | is_sleeping ( ) const                    |
| void       | set_angular_velocity ( Vector3 velocity ) |
| void       | set_linear_velocity ( Vector3 velocity ) |
| void       | set_sleep_state ( bool enabled )         |
| void       | set_transform ( Transform transform )     |

12.220.3 Member Function Description

- void add_force ( Vector3 force, Vector3 pos )
- void apply_impulse ( Vector3 pos, Vector3 j )
- Vector3 get_angular_velocity ( ) const
- RID get_contact_collider ( int contact_idx ) const
- int get_contact_collider_id ( int contact_idx ) const
- Object get_contact_collider_object ( int contact_idx ) const
- Vector3 get_contact_collider_pos ( int contact_idx ) const
- int get_contact_collider_shape ( int contact_idx ) const
- Vector3 get_contact_collider_velocity_at_pos ( int contact_idx ) const
- int get_contact_count ( ) const
- Vector3 get_contact_local_normal ( int contact_idx ) const
• `Vector3 get_contact_local_pos ( int contact_idx ) const`
• `int get_contact_local_shape ( int contact_idx ) const`
• `Vector3 get_inverse_inertia ( ) const`
• `float get_inverse_mass ( ) const`
• `Vector3 get_linear_velocity ( ) const`
• `PhysicsDirectSpaceState get_space_state ( )`
• `float get_step ( ) const`
• `float get_total_angular_damp ( ) const`
• `Vector3 get_total_gravity ( ) const`
• `float get_total_linear_damp ( ) const`
• `Transform get_transform ( ) const`
• `void integrate_forces ( )`
• `bool is_sleeping ( ) const`
• `void set_angular_velocity ( Vector3 velocity )`
• `void set_linear_velocity ( Vector3 velocity )`
• `void set_sleep_state ( bool enabled )`
• `void set_transform ( Transform transform )`

## 12.221 PhysicsDirectBodyStateSW

Inherits: `PhysicsDirectBodyState < Object`

Category: Core

## 12.221.1 Brief Description

## 12.222 PhysicsDirectSpaceState

Inherits: `Object`

Category: Core
12.222.1 Brief Description

12.222.2 Member Functions

<table>
<thead>
<tr>
<th>Array</th>
<th>cast_motion ( PhysicsShapeQueryParameters shape, Vector3 motion )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>collide_shape ( PhysicsShapeQueryParameters shape, int max_results=32 )</td>
</tr>
<tr>
<td>Dictionary</td>
<td>get_rest_info ( PhysicsShapeQueryParameters shape )</td>
</tr>
<tr>
<td>Dictionary</td>
<td>intersect_ray ( Vector3 from, Vector3 to, Array exclude=Array(), int layer_mask=2147483647, int type_mask=15 )</td>
</tr>
<tr>
<td>Array</td>
<td>intersect_shape ( PhysicsShapeQueryParameters shape, int max_results=32 )</td>
</tr>
</tbody>
</table>

12.222.3 Numeric Constants

- TYPE_MASK_STATIC_BODY = 1
- TYPE_MASK_KINEMATIC_BODY = 2
- TYPE_MASK_RIGID_BODY = 4
- TYPE_MASK_CHARACTER_BODY = 8
- TYPE_MASK_AREA = 16
- TYPE_MASK_COLLISION = 15

12.222.4 Member Function Description

- Array cast_motion ( PhysicsShapeQueryParameters shape, Vector3 motion )
- Array collide_shape ( PhysicsShapeQueryParameters shape, int max_results=32 )
- Dictionary get_rest_info ( PhysicsShapeQueryParameters shape )
- Dictionary intersect_ray ( Vector3 from, Vector3 to, Array exclude=Array(), int layer_mask=2147483647, int type_mask=15 )
- Array intersect_shape ( PhysicsShapeQueryParameters shape, int max_results=32 )

12.223 PhysicsServer

Inherits: Object
Inherited By: PhysicsServerSW
Category: Core

12.223.1 Brief Description

12.223.2 Member Functions

| void        | area_add_shape ( RID area, RID shape, Transform transform=Transform() ) |

Continued on next page
Table 12.27 – continued from previous page

<table>
<thead>
<tr>
<th>void</th>
<th>area_attach_object_instance_ID ( RID area, int id )</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>area_clear_shapes ( RID area )</td>
</tr>
<tr>
<td>RID</td>
<td>area_create ()</td>
</tr>
<tr>
<td>int</td>
<td>area_get_object_instance_ID ( RID area ) const</td>
</tr>
<tr>
<td>void</td>
<td>area_get_param ( RID area, int param ) const</td>
</tr>
<tr>
<td>RID</td>
<td>area_get_shape ( RID area, int shape_idx ) const</td>
</tr>
<tr>
<td>int</td>
<td>area_get_shape_count ( RID area ) const</td>
</tr>
<tr>
<td>Transform</td>
<td>area_get_shape_transform ( RID area, int shape_idx ) const</td>
</tr>
<tr>
<td>RID</td>
<td>area_get_space ( RID area ) const</td>
</tr>
<tr>
<td>int</td>
<td>area_get_space_override_mode ( RID area ) const</td>
</tr>
<tr>
<td>bool</td>
<td>area_is_ray_pickable ( RID area ) const</td>
</tr>
<tr>
<td>void</td>
<td>area_remove_shape ( RID area, int shape_idx )</td>
</tr>
<tr>
<td>void</td>
<td>area_set_collision_mask ( RID area, int mask )</td>
</tr>
<tr>
<td>void</td>
<td>area_set_layer_mask ( RID area, int mask )</td>
</tr>
<tr>
<td>void</td>
<td>area_set_monitor_callback ( RID area, Object receiver, String method )</td>
</tr>
<tr>
<td>void</td>
<td>area_set_param ( RID area, int param, Variant value )</td>
</tr>
<tr>
<td>void</td>
<td>area_set_ray_pickable ( RID area, bool enable )</td>
</tr>
<tr>
<td>void</td>
<td>area_set_shape ( RID area, int shape_idx, RID shape )</td>
</tr>
<tr>
<td>void</td>
<td>area_set_shape_transform ( RID area, int shape_idx, Transform transform )</td>
</tr>
<tr>
<td>void</td>
<td>area_set_space ( RID area, RID space )</td>
</tr>
<tr>
<td>void</td>
<td>area_set_space_override_mode ( RID area, int mode )</td>
</tr>
<tr>
<td>void</td>
<td>area_set_transform ( RID area, Transform transform )</td>
</tr>
<tr>
<td>void</td>
<td>body_add_collision_exception ( RID body, RID excepted_body )</td>
</tr>
<tr>
<td>void</td>
<td>body_add_shape ( RID body, RID shape, Transform transform=Transform() )</td>
</tr>
<tr>
<td>void</td>
<td>body_apply_impulse ( RID body, Vector3 pos, Vector3 impulse )</td>
</tr>
<tr>
<td>void</td>
<td>body_attach_object_instance_ID ( RID body, int id )</td>
</tr>
<tr>
<td>void</td>
<td>body_clear_shapes ( RID body )</td>
</tr>
<tr>
<td>RID</td>
<td>body_create ( int mode=2, bool init_sleeping=false )</td>
</tr>
<tr>
<td>int</td>
<td>body_get_axis_lock ( RID body ) const</td>
</tr>
<tr>
<td>int</td>
<td>body_get_collision_mask ( RID body, int arg1 ) const</td>
</tr>
<tr>
<td>int</td>
<td>body_get_layer_mask ( RID body, int arg1 ) const</td>
</tr>
<tr>
<td>int</td>
<td>body_get_max_contacts_reported ( RID body ) const</td>
</tr>
<tr>
<td>int</td>
<td>body_get_mode ( RID body ) const</td>
</tr>
<tr>
<td>int</td>
<td>body_get_object_instance_ID ( RID body ) const</td>
</tr>
<tr>
<td>float</td>
<td>body_get_param ( RID body, int param ) const</td>
</tr>
<tr>
<td>RID</td>
<td>body_get_shape ( RID body, int shape_idx ) const</td>
</tr>
<tr>
<td>int</td>
<td>body_get_shape_count ( RID body ) const</td>
</tr>
<tr>
<td>Transform</td>
<td>body_get_shape_transform ( RID body, int shape_idx ) const</td>
</tr>
<tr>
<td>RID</td>
<td>body_get_space ( RID body ) const</td>
</tr>
<tr>
<td>void</td>
<td>body_is_continuous_collision_detection_enabled ( RID body ) const</td>
</tr>
<tr>
<td>bool</td>
<td>body_is_omitting_force_integration ( RID body ) const</td>
</tr>
<tr>
<td>bool</td>
<td>body_is_ray_pickable ( RID body ) const</td>
</tr>
<tr>
<td>void</td>
<td>body_remove_collision_exception ( RID body, RID excepted_body )</td>
</tr>
<tr>
<td>void</td>
<td>body_remove_shape ( RID body, int shape_idx )</td>
</tr>
<tr>
<td>void</td>
<td>body_set_axis_lock ( RID body, int axis )</td>
</tr>
<tr>
<td>void</td>
<td>body_set_axis_velocity ( RID body, Vector3 axis_velocity )</td>
</tr>
<tr>
<td>void</td>
<td>body_set_collision_mask ( RID body, int mask )</td>
</tr>
<tr>
<td>void</td>
<td>body_set_enable_continuous_collision_detection ( RID body, bool enable )</td>
</tr>
</tbody>
</table>

Continued on next page
void body_set_force_integration_callback ( RID body, Object receiver, String method, Variant userdata=NULL )
void body_set_layer_mask ( RID body, int mask )
void body_set_max_contacts_reported ( RID body, int amount )
void body_set_mode ( RID body, int mode )
void body_set_omit_force_integration ( RID body, bool enable )
void body_set_param ( RID body, int param, float value )
void body_set_ray_pickable ( RID body, bool enable )
void body_set_shape ( RID body, int shape_id, RID shape )
void body_set_shape_transform ( RID body, int shape_id, Transform transform )
void body_set_space ( RID body, RID space )
void body_set_state ( RID body, int state, Variant value )
float cone_twist_joint_get_param ( RID joint, int param ) const
void cone_twist_joint_set_param ( RID joint, int param, float value )
void free_rid ( RID rid )
bool generic_6dof_joint_get_flag ( RID joint, int axis, int flag )
float generic_6dof_joint_get_param ( RID joint, int axis, int param )
void generic_6dof_joint_set_flag ( RID joint, int axis, int flag, bool enable )
void generic_6dof_joint_set_param ( RID joint, int axis, int param, float value )
int get_process_info ( int process_info )
bool hinge_joint_get_flag ( RID joint, int flag ) const
float hinge_joint_get_param ( RID joint, int param ) const
void hinge_joint_set_flag ( RID joint, int flag, bool enabled )
void hinge_joint_set_param ( RID joint, int param, float value )
RID joint_create_cone_twist ( RID body_A, Transform local_ref_A, RID body_B, Transform local_ref_B )
RID joint_create_generic_6dof ( RID body_A, Transform local_ref_A, RID body_B, Transform local_ref_B )
RID joint_create_hinge ( RID body_A, Transform hinge_A, RID body_B, Transform hinge_B )
RID joint_create_pin ( RID body_A, Vector3 local_A, RID body_B, Vector3 local_B )
RID joint_create_slider ( RID body_A, Transform local_ref_A, RID body_B, Transform local_ref_B )
int joint_get_solver_priority ( RID joint ) const
void joint_set_solver_priority ( RID joint, int priority )
Vector3 pin_joint_get_local_A ( RID joint ) const
Vector3 pin_joint_get_local_B ( RID joint ) const
float pin_joint_get_param ( RID joint, int param ) const
void pin_joint_set_local_A ( RID joint, Vector3 local_A )
void pin_joint_set_local_B ( RID joint, Vector3 local_B )
void pin_joint_set_param ( RID joint, int param, float value )
void set_active ( bool active )
RID shape_create ( int type )
void shape_get_data ( RID shape ) const
int shape_get_type ( RID shape ) const
void shape_set_data ( RID shape, Variant data )
float slider_joint_get_param ( RID joint, int param ) const
void slider_joint_set_param ( RID joint, int param, float value )
RID space_create ()
PhysicsDirectSpaceState space_get_direct_state ( RID space )
float space_get_param ( RID space, int param ) const
bool space_is_active ( RID space ) const
void space_set_active ( RID space, bool active )
void space_set_param ( RID space, int param, float value )
### 12.223.3 Numeric Constants

- JOINT_PIN = 0
- JOINT_HINGE = 1
- JOINT_SLIDER = 2
- JOINT_CONE_TWIST = 3
- JOINT_6DOF = 4
- PIN_JOINT_BIAS = 0
- PIN_JOINT_DAMPING = 1
- PIN_JOINT_IMPULSE_CLAMP = 2
- HINGE_JOINT_BIAS = 0
- HINGE_JOINT_LIMIT_UPPER = 1
- HINGE_JOINT_LIMIT_LOWER = 2
- HINGE_JOINT_LIMIT_BIAS = 3
- HINGE_JOINT_LIMIT_SOFTNESS = 4
- HINGE_JOINT_LIMIT_RELAXATION = 5
- HINGE_JOINT_MOTOR_TARGET_VELOCITY = 6
- HINGE_JOINT_MOTOR_MAX_IMPULSE = 7
- HINGE_JOINT_FLAG_USE_LIMIT = 0
- HINGE_JOINT_FLAG_ENABLE_MOTOR = 1
- SLIDER_JOINT_LINEAR_LIMIT_UPPER = 0
- SLIDER_JOINT_LINEAR_LIMIT_LOWER = 1
- SLIDER_JOINT_LINEAR_LIMIT_SOFTNESS = 2
- SLIDER_JOINT_LINEAR_LIMIT_RESTITUTION = 3
- SLIDER_JOINT_LINEAR_LIMIT_DAMPING = 4
- SLIDER_JOINT_LINEAR_MOTION_SOFTNESS = 5
- SLIDER_JOINT_LINEAR_MOTION_RESTITUTION = 6
- SLIDER_JOINT_LINEAR_MOTION_DAMPING = 7
- SLIDER_JOINT_LINEAR_ORTHOGONAL_SOFTNESS = 8
- SLIDER_JOINT_LINEAR_ORTHOGONAL_RESTITUTION = 9
- SLIDER_JOINT_LINEAR_ORTHOGONAL_DAMPING = 10
- SLIDER_JOINT_ANGULAR_LIMIT_UPPER = 11
- SLIDER_JOINT_ANGULAR_LIMIT_LOWER = 12
- SLIDER_JOINT_ANGULAR_LIMIT_SOFTNESS = 13
- SLIDER_JOINT_ANGULAR_LIMIT_RESTITUTION = 14
- SLIDER_JOINT_ANGULAR_LIMIT_DAMPING = 15
- SLIDER_JOINT_ANGULAR_MOTION_SOFTNESS = 16
• SLIDER_JOINT_ANGULAR_MOTION_RESTITUTION = 17
• SLIDER_JOINT_ANGULAR_MOTION_DAMPING = 18
• SLIDER_JOINT_ANGULAR_ORTHOGONAL_SOFTNESS = 19
• SLIDER_JOINT_ANGULAR_ORTHOGONAL_RESTITUTION = 20
• SLIDER_JOINT_ANGULAR_ORTHOGONAL_DAMPING = 21
• SLIDER_JOINT_MAX = 22
• CONE_TWIST_JOINT_SWING_SPAN = 0
• CONE_TWIST_JOINT_TWIST_SPAN = 1
• CONE_TWIST_JOINT_BIAS = 2
• CONE_TWIST_JOINT_SOFTNESS = 3
• CONE_TWIST_JOINT_RELAXATION = 4
• G6DOF_JOINT_LINEAR_LOWER_LIMIT = 0
• G6DOF_JOINT_LINEAR_UPPER_LIMIT = 1
• G6DOF_JOINT_LINEAR_LIMIT_SOFTNESS = 2
• G6DOF_JOINT_LINEAR_RESTITUTION = 3
• G6DOF_JOINT_LINEAR_DAMPING = 4
• G6DOF_JOINT_ANGULAR_LOWER_LIMIT = 5
• G6DOF_JOINT_ANGULAR_UPPER_LIMIT = 6
• G6DOF_JOINT_ANGULAR_LIMIT_SOFTNESS = 7
• G6DOF_JOINT_ANGULAR_DAMPING = 8
• G6DOF_JOINT_ANGULAR_RESTITUTION = 9
• G6DOF_JOINT_ANGULAR_FORCE_LIMIT = 10
• G6DOF_JOINT_ANGULAR_ERP = 11
• G6DOF_JOINT_ANGULAR_MOTOR_TARGET VELOCITY = 12
• G6DOF_JOINT_ANGULAR_MOTOR_FORCE_LIMIT = 13
• G6DOF_JOINT_FLAG_ENABLE_LINEAR_LIMIT = 0
• G6DOF_JOINT_FLAG_ENABLE_ANGULAR_LIMIT = 1
• G6DOF_JOINT_FLAG_ENABLE_MOTOR = 2
• SHAPE_PLANE = 0
• SHAPE_RAY = 1
• SHAPE_SPHERE = 2
• SHAPE_BOX = 3
• SHAPE_CAPSULE = 4
• SHAPE_CONVEX_POLYGON = 5
• SHAPE_CONCAVE_POLYGON = 6
• SHAPE_HEIGHTMAP = 7
• SHAPE_CUSTOM = 8
• AREA_PARAM_GRAVITY = 0
• AREA_PARAM_GRAVITY_VECTOR = 1
• AREA_PARAM_GRAVITY_IS_POINT = 2
• AREA_PARAM_GRAVITY_DISTANCE_SCALE = 3
• AREA_PARAM_GRAVITY_POINT_ATTENUATION = 4
• AREA_PARAM_LINEAR_DAMP = 5
• AREA_PARAM_ANGULAR_DAMP = 6
• AREA_PARAM_PRIORITY = 7
• AREA_SPACE_OVERRIDE_DISABLED = 0 — This area does not affect gravity/damp. These are generally areas that exist only to detect collisions, and objects entering or exiting them.
• AREA_SPACE_OVERRIDE_COMBINE = 1 — This area adds its gravity/damp values to whatever has been calculated so far. This way, many overlapping areas can combine their physics to make interesting effects.
• AREA_SPACE_OVERRIDE_COMBINE_REPLACE = 2 — This area adds its gravity/damp values to whatever has been calculated so far. Then stops taking into account the rest of the areas, even the default one.
• AREA_SPACE_OVERRIDE_REPLACE = 3 — This area replaces any gravity/damp, even the default one, and stops taking into account the rest of the areas.
• AREA_SPACE_OVERRIDE_REPLACE_COMBINE = 4 — This area replaces any gravity/damp calculated so far, but keeps calculating the rest of the areas, down to the default one.
• BODY_MODE_STATIC = 0
• BODY_MODE_KINEMATIC = 1
• BODY_MODE_RIGID = 2
• BODY_MODE_CHARACTER = 3
• BODY_PARAM_BOUNCE = 0
• BODY_PARAM_FRICTION = 1
• BODY_PARAM_MASS = 2
• BODY_PARAM_GRAVITY_SCALE = 3
• BODY_PARAM_ANGULAR_DAMP = 5
• BODY_PARAM_LINEAR_DAMP = 4
• BODY_PARAM_MAX = 6
• BODY_STATE_TRANSFORM = 0
• BODY_STATE_LINEAR_VELOCITY = 1
• BODY_STATE_ANGULAR_VELOCITY = 2
• BODY_STATE_SLEEPING = 3
• BODY_STATE_CAN_SLEEP = 4
• AREA_BODY_ADDED = 0
• AREA_BODY_REMOVED = 1
• INFO_ACTIVE_OBJECTS = 0
• INFO_COLLISION_PAIRS = 1
• INFO_ISLAND_COUNT = 2

12.223.4 Member Function Description

• void area_add_shape ( RID area, RID shape, Transform transform=Transform() )
• void area_attach_object_instance_ID ( RID area, int id )
• void area_clear_shapes ( RID area )
• RID area_create ( )
• int area_get_object_instance_ID ( RID area ) const
• void area_get_param ( RID area, int param ) const
• RID area_get_shape ( RID area, int shape_idx ) const
• int area_get_shape_count ( RID area ) const
• Transform area_get_shape_transform ( RID area, int shape_idx ) const
• RID area_get_space ( RID area ) const
• int area_get_space_override_mode ( RID area ) const
• Transform area_get_transform ( RID area ) const
• bool area_is_ray_pickable ( RID area ) const
• void area_remove_shape ( RID area, int shape_idx )
• void area_set_collision_mask ( RID area, int mask )
• void area_set_layer_mask ( RID area, int mask )
• void area_set_monitor_callback ( RID area, Object receiver, String method )
• void area_set_param ( RID area, int param, Variant value )
• void area_set_ray_pickable ( RID area, bool enable )
• void area_set_shape ( RID area, int shape_idx, RID shape )
• void area_set_shape_transform ( RID area, int shape_idx, Transform transform )
• void area_set_space ( RID area, RID space )
• void area_set_space_override_mode ( RID area, int mode )
• void area_set_transform ( RID area, Transform transform )
• void body_add_collision_exception ( RID body, RID excepted_body )
• void body_add_shape ( RID body, RID shape, Transform transform=Transform() )
• void body_apply_impulse ( RID body, Vector3 pos, Vector3 impulse )
• void body_attach_object_instance_ID ( RID body, int id )
• void body_clear_shapes ( RID body )
• RID body_create ( int mode=2, bool init_sleeping=false )
• int body_get_axis_lock ( RID body ) const
• int body_get_collision_mask ( RID body, int arg1 ) const
• int body_get_layer_mask ( RID body, int arg1 ) const
• int body_get_max_contacts_reported ( RID body ) const
• int body_get_mode ( RID body ) const
• int body_get_object_instance_ID ( RID body ) const
• float body_get_param ( RID body, int param ) const
• RID body_get_shape ( RID body, int shape_idx ) const
• int body_get_shape_count ( RID body ) const
• Transform body_get_shape_transform ( RID body, int shape_idx ) const
• RID body_get_space ( RID body ) const
• void body_get_state ( RID body, int state ) const
• bool body_is_continuous_collision_detection_enabled ( RID body ) const
• bool body_is_omitting_force_integration ( RID body ) const
• bool body_is_ray_pickable ( RID body ) const
• void body_remove_collision_exception ( RID body, RID excepted_body )
• void body_remove_shape ( RID body, int shape_idx )
• void body_set_axis_lock ( RID body, int axis )
• void body_set_axis_velocity ( RID body, Vector3 axis_velocity )
• void body_set_collision_mask ( RID body, int mask )
• void body_set_enable_continuous_collision_detection ( RID body, bool enable )
• void body_set_force_integration_callback ( RID body, Object receiver, String method, Variant userdata=NULL )
• void body_set_layer_mask ( RID body, int mask )
• void body_set_max_contacts_reported ( RID body, int amount )
• void body_set_mode ( RID body, int mode )
• void body_set_omit_force_integration ( RID body, bool enable )
• void body_set_param ( RID body, int param, float value )
• void body_set_ray_pickable ( RID body, bool enable )
• void body_set_shape ( RID body, int shape_idx, RID shape )
• void body_set_shape_transform ( RID body, int shape_idx, Transform transform )
• void body_set_space ( RID body, RID space )
• void body_set_state ( RID body, int state, Variant value )
• float cone_twist_joint_get_param ( RID joint, int param ) const
• void cone_twist_joint_set_param ( RID joint, int param, float value )
• void free_rid ( RID rid )
• bool generic_6dof_joint_get_flag ( RID joint, int axis, int flag )
• float generic_6dof_joint_get_param ( RID joint, int axis, int param )
• void generic_6dof_joint_set_flag ( RID joint, int axis, int flag, bool enable )
• void generic_6dof_joint_set_param ( RID joint, int axis, int param, float value )
• int get_process_info ( int process_info )
• bool hinge_joint_get_flag ( RID joint, int flag ) const
• float hinge_joint_get_param ( RID joint, int param ) const
• void hinge_joint_set_flag ( RID joint, int flag, bool enabled )
• void hinge_joint_set_param ( RID joint, int param, float value )
• RID joint_create_cone_twist ( RID body_A, Transform local_ref_A, RID body_B, Transform local_ref_B )
• RID joint_create_generic_6dof ( RID body_A, Transform local_ref_A, RID body_B, Transform local_ref_B )
• RID joint_create_hinge ( RID body_A, Transform hinge_A, RID body_B, Transform hinge_B )
• RID joint_create_pin ( RID body_A, Vector3 local_A, RID body_B, Vector3 local_B )
• RID joint_create_slider ( RID body_A, Transform local_ref_A, RID body_B, Transform local_ref_B )
• int joint_get_solver_priority ( RID joint ) const
• int joint_get_type ( RID joint ) const
• void joint_set_solver_priority ( RID joint, int priority )
• Vector3 pin_joint_get_local_A ( RID joint ) const
• Vector3 pin_joint_get_local_B ( RID joint ) const
• float pin_joint_get_param ( RID joint, int param ) const
• void pin_joint_set_local_A ( RID joint, Vector3 local_A )
• void pin_joint_set_local_B ( RID joint, Vector3 local_B )
• void pin_joint_set_param ( RID joint, int param, float value )
• void set_active ( bool active )
• RID shape_create ( int type )
• void shape_get_data ( RID shape ) const
• int shape_get_type ( RID shape ) const
• void shape_set_data ( RID shape, Variant data )
• float slider_joint_get_param ( RID joint, int param ) const
• void slider_joint_set_param ( RID joint, int param, float value )
• RID space_create ( )
• PhysicsDirectSpaceState space_get_direct_state ( RID space )
• float space_get_param ( RID space, int param ) const
• bool space_is_active ( RID space ) const
• void space_set_active ( RID space, bool active )
• void space_set_param ( RID space, int param, float value )
12.224 PhysicsServerSW

Inherits: PhysicsServer < Object
Category: Core

12.224.1 Brief Description

12.225 PhysicsShapeQueryParameters

Inherits: Reference < Object
Category: Core

12.225.1 Brief Description

12.225.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>get_exclude() const</td>
</tr>
<tr>
<td>int</td>
<td>get_layer_mask() const</td>
</tr>
<tr>
<td>float</td>
<td>get_margin() const</td>
</tr>
<tr>
<td>int</td>
<td>get_object_type_mask() const</td>
</tr>
<tr>
<td>RID</td>
<td>get_shape_rid() const</td>
</tr>
<tr>
<td>Transform</td>
<td>get_transform() const</td>
</tr>
<tr>
<td>void</td>
<td>set_exclude(Array exclude)</td>
</tr>
<tr>
<td>void</td>
<td>set_layer_mask(int layer_mask)</td>
</tr>
<tr>
<td>void</td>
<td>set_margin(float margin)</td>
</tr>
<tr>
<td>void</td>
<td>set_object_type_mask(int object_type_mask)</td>
</tr>
<tr>
<td>void</td>
<td>set_shape(Shape shape)</td>
</tr>
<tr>
<td>void</td>
<td>set_shape_rid(RID shape)</td>
</tr>
<tr>
<td>void</td>
<td>set_transform(Transform transform)</td>
</tr>
</tbody>
</table>

12.225.3 Member Function Description

- Array get_exclude() const
- int get_layer_mask() const
- float get_margin() const
- int get_object_type_mask() const
- RID get_shape_rid() const
- Transform get_transform() const
- void set_exclude(Array exclude)
- void set_layer_mask(int layer_mask)
- void set_margin(float margin)
- void set_object_type_mask(int object_type_mask)
- void set_shape(Shape shape)
• void set_shape_rid ( RID shape )
• void set_transform ( Transform transform )

12.226 PhysicsShapeQueryResult

Inherits: Reference < Object
Category: Core

12.226.1 Brief Description

Result of a shape query in Physics2DServer.

12.226.2 Member Functions

<table>
<thead>
<tr>
<th>int</th>
<th>get_result_count ( ) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>get_result_object ( int idx ) const</td>
</tr>
<tr>
<td>int</td>
<td>get_result_object_id ( int idx ) const</td>
</tr>
<tr>
<td>int</td>
<td>get_result_object_shape ( int idx ) const</td>
</tr>
<tr>
<td>RID</td>
<td>get_result_rid ( int idx ) const</td>
</tr>
</tbody>
</table>

12.226.3 Member Function Description

• int get_result_count ( ) const
• Object get_result_object ( int idx ) const
• int get_result_object_id ( int idx ) const
• int get_result_object_shape ( int idx ) const
• RID get_result_rid ( int idx ) const

12.227 PinJoint

Inherits: Joint < Spatial < Node < Object
Category: Core

12.227.1 Brief Description

12.227.2 Member Functions

<table>
<thead>
<tr>
<th>float</th>
<th>get_param ( int param ) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>set_param ( int param, float value )</td>
</tr>
</tbody>
</table>
12.227.3 Numeric Constants

- PARAM_BIAS = 0
- PARAM_DAMPING = 1
- PARAM_IMPULSE_CLAMP = 2

12.227.4 Member Function Description

- float get_param ( int param ) const
- void set_param ( int param, float value )

12.228 PinJoint2D

Inherits: Joint2D < Node2D < CanvasItem < Node < Object

Category: Core

12.228.1 Brief Description

Pin Joint for 2D Shapes.

12.228.2 Member Functions

<table>
<thead>
<tr>
<th>float</th>
<th>get_softness ( ) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>set_softness ( float softness )</td>
</tr>
</tbody>
</table>

12.228.3 Description

Pin Joint for 2D Rigid Bodies. It pins 2 bodies (rigid or static) together, or a single body to a fixed position in space.

12.228.4 Member Function Description

- float get_softness ( ) const
- void set_softness ( float softness )

12.229 Plane

Category: Built-In Types

12.229.1 Brief Description

Plane in hessian form.
### 12.229.2 Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plane (float a, float b, float c, float d)</td>
<td>Creates a plane from the three parameters “a”, “b”, “c” and “d”.</td>
</tr>
<tr>
<td>Plane (Vector3 v1, Vector3 v2, Vector3 v3)</td>
<td>Creates a plane from three points.</td>
</tr>
<tr>
<td>Plane (Vector3 normal, float d)</td>
<td>Creates a plane from the normal and the plane’s distance to the origin.</td>
</tr>
</tbody>
</table>

- `Vector3 center()` Returns the center of the plane.
- `float distance_to(Vector3 point)` Returns the shortest distance from the plane to the position “point”.
- `Vector3 get_any_point()`
- `bool has_point(Vector3 point, float epsilon=0.00001)`
- `Vector3 intersect_3(Plane b, Plane c)`
- `Vector3 intersects_ray(Vector3 from, Vector3 dir)`
- `Vector3 intersects_segment(Vector3 begin, Vector3 end)`
- `bool is_point_over(Vector3 point)`
- `Plane normalized()`
- `Vector3 project(Vector3 point)`

### 12.229.3 Member Variables

- `float d`
- `Vector3 normal`
- `float x`
- `float y`
- `float z`

### 12.229.4 Description

Plane represents a normalized plane equation. Basically, “normal” is the normal of the plane (a,b,c normalized), and “d” is the distance from the origin to the plane (in the direction of “normal”). “Over” or “Above” the plane is considered the side of the plane towards where the normal is pointing.

### 12.229.5 Member Function Description

- `Plane Plane (float a, float b, float c, float d)` Creates a plane from the three parameters “a”, “b”, “c” and “d”.
- `Plane Plane (Vector3 v1, Vector3 v2, Vector3 v3)` Creates a plane from three points.
- `Plane Plane (Vector3 normal, float d)` Creates a plane from the normal and the plane’s distance to the origin.
- `Vector3 center()` Returns the center of the plane.
- `float distance_to(Vector3 point)` Returns the shortest distance from the plane to the position “point”.
- `Vector3 get_any_point()`
Returns a point on the plane.

- `bool has_point ( Vector3 point, float epsilon=0.00001 )`
  Returns true if “point” is inside the plane (by a very minimum threshold).

- `Vector3 intersect_3 ( Plane b, Plane c )`
  Returns the intersection point of the three planes “b”, “c” and this plane. If no intersection is found null is returned.

- `Vector3 intersects_ray ( Vector3 from, Vector3 dir )`
  Returns the intersection point of a ray consisting of the position “from” and the direction normal “dir” with this plane. If no intersection is found null is returned.

- `Vector3 intersects_segment ( Vector3 begin, Vector3 end )`
  Returns the intersection point of a segment from position “begin” to position “end” with this plane. If no intersection is found null is returned.

- `bool is_point_over ( Vector3 point )`
  Returns true if “point” is located above the plane.

- `Plane normalized ( )`
  Returns a copy of the plane, normalized.

- `Vector3 project ( Vector3 point )`
  Returns the orthogonal projection of point “p” into a point in the plane.

### 12.230 PlaneShape

**Inherits:** `Shape < Resource < Reference < Object`

**Category:** Core

#### 12.230.1 Brief Description

#### 12.230.2 Member Functions

<table>
<thead>
<tr>
<th>Plane</th>
<th>get_plane ( ) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>set_plane ( Plane plane )</td>
</tr>
</tbody>
</table>

#### 12.230.3 Member Function Description

- `Plane get_plane ( ) const`
- `void set_plane ( Plane plane )`

### 12.231 Polygon2D

**Inherits:** `Node2D < CanvasItem < Node < Object`

**Category:** Core
12.231.1 Brief Description

2D polygon representation

12.231.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color</td>
<td>get_color() const</td>
</tr>
<tr>
<td>bool</td>
<td>get_invert() const</td>
</tr>
<tr>
<td>float</td>
<td>get_invert_border() const</td>
</tr>
<tr>
<td>Vector2</td>
<td>get_offset() const</td>
</tr>
<tr>
<td>Vector2Array</td>
<td>get_polygon() const</td>
</tr>
<tr>
<td>Object</td>
<td>get_texture() const</td>
</tr>
<tr>
<td>Vector2</td>
<td>get_texture_offset() const</td>
</tr>
<tr>
<td>float</td>
<td>get_texture_rotation() const</td>
</tr>
<tr>
<td>Vector2</td>
<td>get_texture_scale() const</td>
</tr>
<tr>
<td>Vector2Array</td>
<td>get_uv() const</td>
</tr>
<tr>
<td>ColorArray</td>
<td>get_vertex_colors() const</td>
</tr>
<tr>
<td>void</td>
<td>set_color(Color color)</td>
</tr>
<tr>
<td>void</td>
<td>set_invert(bool invert)</td>
</tr>
<tr>
<td>void</td>
<td>set_invert_border(float invert_border)</td>
</tr>
<tr>
<td>void</td>
<td>set_offset(Vector2 offset)</td>
</tr>
<tr>
<td>void</td>
<td>set_polygon(Vector2Array polygon)</td>
</tr>
<tr>
<td>void</td>
<td>set_texture(Object texture)</td>
</tr>
<tr>
<td>void</td>
<td>set_texture_offset(Vector2 texture_offset)</td>
</tr>
<tr>
<td>void</td>
<td>set_texture_rotation(float texture_rotation)</td>
</tr>
<tr>
<td>void</td>
<td>set_texture_scale(Vector2 texture_scale)</td>
</tr>
<tr>
<td>void</td>
<td>set_uv(Vector2Array uv)</td>
</tr>
<tr>
<td>void</td>
<td>set_vertex_colors(ColorArray vertex_colors)</td>
</tr>
</tbody>
</table>

12.231.3 Description

A Polygon2D is defined by a set of n vertices connected together by line segments, meaning that the vertex 1 will be connected with vertex 2, vertex 2 with vertex 3 ..., vertex n-1 with vertex n and vertex n with vertex 1 in order to close the loop and define a polygon.

12.231.4 Member Function Description

- **Color get_color() const**
  Return the polygon fill color.

- **bool get_invert() const**
  Return whether this polygon is inverted or not.

- **float get_invert_border() const**
  Return the added padding around the bounding box.

- **Vector2 get_offset() const**
  Return the offset for the polygon vertices.

- **Vector2Array get_polygon() const**
Return the set of vertices that defines this polygon.

- `Object get_texture ( ) const`

Return the polygon texture

- `Vector2 get_texture_offset ( ) const`

Return the polygon texture offset.

- `float get_texture_rotation ( ) const`

Return the rotation in radians of the texture polygon.

- `Vector2 get_texture_scale ( ) const`

Return the uv coordinate multiplier.

- `Vector2Array get_uv ( ) const`

Return the texture coordinates associated with every vertex of the polygon.

- `ColorArray get_vertex_colors ( ) const`

Return the list of vertex colors.

- `void set_color ( Color color )`

Set the polygon fill color. If the polygon has a texture defined, the defined texture will be multiplied by the polygon fill color. This, also, is the default color for those vertices that are not defined by `get_vertex_colors`.

- `void set_invert ( bool invert )`

Set the polygon as the defined polygon bounding box minus the defined polygon (the defined polygon will appear as a hole on the square that contains the defined polygon).

- `void set_invert_border ( float invert_border )`

Add extra padding around the bounding box, making it bigger. Too small a value can make the polygon triangulate strangely, due to numerical imprecision.

- `void set_offset ( Vector2 offset )`

Set the an offset that will be added to the vertices’ position. E.g. if the offset is set to (10,10) then all the polygon points will move 10 units to the right and 10 units to the bottom.

- `void set_polygon ( Vector2Array polygon )`

Define the set of vertices that will represent the polygon.

- `void set_texture ( Object texture )`

Set the polygon texture.

- `void set_texture_offset ( Vector2 texture_offset )`

Set the offset of the polygon texture. Initially the texture will appear anchored to the polygon position, the offset is used to move the texture location away from that point (notice that the texture origin is set to its top left corner, so when offset is 0,0 the top left corner of the texture is at the polygon position), for example setting the offset to 10, 10 will move the texture 10 units to the left and 10 units to the top.

- `void set_texture_rotation ( float texture_rotation )`

Set the amount of rotation of the polygon texture, `texture_rotation` is specified in radians and clockwise rotation.

- `void set_texture_scale ( Vector2 texture_scale )`
Set the value that will multiply the uv coordinates (get_uv) when applying the texture. Larger values make the texture smaller, and vice versa.

- void set_uv (Vector2Array uv)

Set the texture coordinates for every vertex of the polygon. There should be one uv vertex for every vertex in the polygon. If there are less, the undefined ones will be assumed to be (0,0). Extra uv vertices are ignored.

- void set_vertex_colors (ColorArray vertex_colors)

Set the color for each vertex of the polygon. There should be one color for every vertex in the polygon. If there are less, the undefined ones will be assumed to be get_color. Extra color entries are ignored.

Colors are interpolated between vertices, resulting in smooth gradients when they differ.

### 12.23.2 PolygonPathFinder

Inherits: Resource < Reference < Object

Category: Core

#### 12.23.2.1 Brief Description

#### 12.23.2.2 Member Functions

<table>
<thead>
<tr>
<th>Vector2Array</th>
<th>find_path (Vector2 from, Vector2 to)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rect2</td>
<td>get_bounds ( ) const</td>
</tr>
<tr>
<td>Vector2</td>
<td>get_closest_point (Vector2 point) const</td>
</tr>
<tr>
<td>Vector2Array</td>
<td>get_intersections (Vector2 from, Vector2 to) const</td>
</tr>
<tr>
<td>float</td>
<td>get_point_penalty (int idx) const</td>
</tr>
<tr>
<td>bool</td>
<td>is_point_inside (Vector2 point) const</td>
</tr>
<tr>
<td>void</td>
<td>set_point_penalty (int idx, float penalty)</td>
</tr>
<tr>
<td>void</td>
<td>setup (Vector2Array points, IntArray connections)</td>
</tr>
</tbody>
</table>

#### 12.23.2.3 Member Function Description

- Vector2Array find_path (Vector2 from, Vector2 to)
- Rect2 get_bounds ( ) const
- Vector2 get_closest_point (Vector2 point) const
- Vector2Array get_intersections (Vector2 from, Vector2 to) const
- float get_point_penalty (int idx) const
- bool is_point_inside (Vector2 point) const
- void set_point_penalty (int idx, float penalty)
- void setup (Vector2Array points, IntArray connections)
12.233 Popup

Inherits: Control < CanvasItem < Node < Object

Inherited By: PopupPanel, PopupDialog, PopupMenu, WindowDialog

Category: Core

12.233.1 Brief Description

Base container control for popups and dialogs.

12.233.2 Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool is_exclusive() const</td>
<td>Returns whether the popup will hide other popups when shown on the screen.</td>
</tr>
<tr>
<td>void popup()</td>
<td>Popup (show the control in modal form).</td>
</tr>
<tr>
<td>void popup_centered(Vector2 size=Vector2(0, 0))</td>
<td></td>
</tr>
<tr>
<td>void popup_centered_minsize(Vector2 minsize=Vector2(0, 0))</td>
<td></td>
</tr>
<tr>
<td>void popup_centered_ratio(float ratio=0.75)</td>
<td></td>
</tr>
<tr>
<td>void set_exclusive(bool enable)</td>
<td></td>
</tr>
</tbody>
</table>

12.233.3 Signals

- about_to_show()
  This signal is emitted when a popup is about to be shown. (often used in PopupMenu for clearing the list of options and creating a new one according to the current context).

- popup_hide()
  This signal is emitted when a popup is hidden.

12.233.4 Numeric Constants

- NOTIFICATION_POST_POPUP = 80 — Notification sent right after the popup is shown.
- NOTIFICATION_POPUP_HIDE = 81 — Notification sent right after the popup is hidden.

12.233.5 Description

Popup is a base Control used to show dialogs and popups. It’s a subwindow and modal by default (see Control) and has helpers for custom popup behavior.

12.233.6 Member Function Description

- bool is_exclusive() const
  Returns whether the popup will hide other popups when shown on the screen.

- void popup()
  Popup (show the control in modal form).
• void popup_centered ( Vector2 size=Vector2(0, 0) )

Popup (show the control in modal form) in the center of the screen, at the current size, or at a size determined by “size”.

• void popup_centered_minsize ( Vector2 minsize=Vector2(0, 0) )

Popup (show the control in modal form) in the center of the screen, ensuring the size is never smaller than minsize.

• void popup_centered_ratio ( float ratio=0.75 )

Popup (show the control in modal form) in the center of the screen, scaled at a ratio of size of the screen.

• void set-exclusive ( bool enable )

Make the popup hide other popups when shown on the screen.

12.234 PopupDialog

Inherits: Popup < Control < CanvasItem < Node < Object

Category: Core

12.234.1 Brief Description

Base class for Popup Dialogs.

12.234.2 Description

PopupDialog is a base class for popup dialogs, along with WindowDialog.

12.235 PopupMenu

Inherits: Popup < Control < CanvasItem < Node < Object

Category: Core

12.235.1 Brief Description

PopupMenu displays a list of options.

12.235.2 Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void add_check_item ( String label, int id=-1, int accel=0 )</td>
<td></td>
</tr>
<tr>
<td>void add_check_shortcut ( ShortCut shortcut, int id=-1 )</td>
<td></td>
</tr>
<tr>
<td>void add_icon_check_item ( Object texture, String label, int id=-1, int accel=0 )</td>
<td></td>
</tr>
<tr>
<td>void add_icon_check_shortcut ( Object texture, ShortCut shortcut, int id=-1 )</td>
<td></td>
</tr>
<tr>
<td>void add_icon_item ( Object texture, String label, int id=-1, int accel=0 )</td>
<td></td>
</tr>
<tr>
<td>void add_icon_shortcut ( Object texture, ShortCut shortcut, int id=-1 )</td>
<td></td>
</tr>
</tbody>
</table>
### 12.235.3 Signals

- **item_pressed ( int ID )**

  This even is emitted when an item is pressed or its accelerator is activated. The id of the item is returned if it exists, else the index.

### 12.235.4 Description

PopupMenu is the typical Control that displays a list of options. They are popular in toolbars or context menus.

### 12.235.5 Member Function Description

- **void add_check_item ( String label, int id=-1, int accel=0 )**
Add a new checkable item with text “label”. An id can optionally be provided, as well as an accelerator. If no id is provided, one will be created from the index. Note that checkable items just display a checkmark, but don’t have any built-in checking behavior and must be checked/unchecked manually.

- void add_check_shortcut (ShortCut shortcut, int id=-1)
- void add_icon_check_item (Object texture, String label, int id=-1, int accel=0)

Add a new checkable item with text “label” and icon “texture”. An id can optionally be provided, as well as an accelerator. If no id is provided, one will be created from the index. Note that checkable items just display a checkmark, but don’t have any built-in checking behavior and must be checked/unchecked manually.

- void add_icon_check_shortcut (Object texture, ShortCut shortcut, int id=-1)
- void add_icon_item (Object texture, String label, int id=-1, int accel=0)

Add a new item with text “label” and icon “texture”. An id can optionally be provided, as well as an accelerator keybinding. If no id is provided, one will be created from the index.

- void add_item (String label, int id=-1, int accel=0)
- void add_separator ()

Add a separator between items. Separators also occupy an index.

- void add_shortcut (ShortCut shortcut, int id=-1)
- void add_submenu_item (String label, String submenu, int id=-1)

Adds an item with a submenu. The submenu is the name of a child PopupMenu node that would be shown when the item is clicked. An id can optionally be provided, but if is isn’t provided, one will be created from the index.

- void clear ()

Clear the popup menu, in effect removing all items.

- int get_item_ID (int idx) const
  Return the id of the item at index “idx”.

- int get_item_accelerator (int idx) const
  Return the accelerator of the item at index “idx”. Accelerators are special combinations of keys that activate the item, no matter which control is focused.

- int get_item_count () const
  Return the amount of items.

- Object get_item_icon (int idx) const
  Return the icon of the item at index “idx”.

- int get_item_index (int id) const
  Find and return the index of the item containing a given id.

- void get_item_metadata (int idx) const
  Return the metadata of an item, which might be of any type. You can set it with set_item_metadata, which provides a simple way of assigning context data to items.
• **String get_item_submenu ( int idx )** const
Return the submenu name of the item at index “idx”.

• **String get_item_text ( int idx )** const
Return the text of the item at index “idx”.

• **String get_item_tooltip ( int idx )** const

• **bool is_hide_on_item_selection ( )**
Returns a boolean that indicates whether or not the PopupMenu will hide on item selection.

• **bool is_item_checkable ( int idx )** const
Return whether the item at index “idx” has a checkbox. Note that checkable items just display a checkmark, but don’t have any built-in checking behavior and must be checked/unchecked manually.

• **bool is_item_checked ( int idx )** const
Return the checkstate status of the item at index “idx”.

• **bool is_item_disabled ( int idx )** const
Return whether the item at index “idx” is disabled. When it is disabled it can’t be selected, or its action invoked.

• **bool is_item_separator ( int idx )** const
Return whether the item is a seperator. If it is, it would be displayed as a line.

• **void remove_item ( int idx )**
Removes the item at index “idx” from the menu. Note that the indexes of items after the removed item are going to be shifted by one.

• **void set_hide_on_item_selection ( bool enable )**
Sets whether or not the PopupMenu will hide on item selection.

• **void set_item_ID ( int idx, int id )**
Set the id of the item at index “idx”.

• **void set_item_accelerator ( int idx, int accel )**
Set the accelerator of the item at index “idx”. Accelerators are special combinations of keys that activate the item, no matter which control is focused.

• **void set_item_as_checkable ( int idx, bool enable )**
Set whether the item at index “idx” has a checkbox. Note that checkable items just display a checkmark, but don’t have any built-in checking behavior and must be checked/unchecked manually.

• **void set_item_as_separator ( int idx, bool enable )**
Mark the item at index “idx” as a seperator, which means that it would be displayed as a mere line.

• **void set_item_checked ( int idx, bool checked )**
Set the checkstate status of the item at index “idx”.

• **void set_item_disabled ( int idx, bool disabled )**
Sets whether the item at index “idx” is disabled or not. When it is disabled it can’t be selected, or its action invoked.

• **void set_item_icon ( int idx, Object icon )**
Set the icon of the item at index “idx”.

12.235. PopupMenu
Godot Engine Documentation, Release 2.1

- void `set_item_metadata` (int `idx`, Variant `metadata`)
  Sets the metadata of an item, which might be of any type. You can later get it with `get_item_metadata`, which provides a simple way of assigning context data to items.

- void `set_item_shortcut` (int `idx`, ShortCut `shortcut`)
- void `set_item_submenu` (int `idx`, String `submenu`)
  Sets the submenu of the item at index “`idx`”. The submenu is the name of a child PopupMenu node that would be shown when the item is clicked.

- void `set_item_text` (int `idx`, String `text`)
  Set the text of the item at index “`idx`”.

- void `set_item_tooltip` (int `idx`, String `tooltip`)
- void `toggle_item_checked` (int `idx`)

12.236 PopupPanel

**Inherits:** Popup < Control < CanvasItem < Node < Object

**Category:** Core

12.236.1 Brief Description

Class for displaying popups with a panel background.

12.236.2 Description

Class for displaying popups with a panel background. In some cases it might be simpler to use than `Popup`, since it provides a configurable background. If you are making windows, better check `WindowDialog`.

12.237 Portal

**Inherits:** VisualInstance < Spatial < Node < Object

**Category:** Core

12.237.1 Brief Description

Portals provide virtual openings to rooms.
### 12.237.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
<th>Const</th>
</tr>
</thead>
<tbody>
<tr>
<td>float</td>
<td><code>get_connect_range()</code></td>
<td>const</td>
</tr>
<tr>
<td>float</td>
<td><code>get_disable_distance()</code></td>
<td>const</td>
</tr>
<tr>
<td>Color</td>
<td><code>get_disabled_color()</code></td>
<td>const</td>
</tr>
<tr>
<td>Vector2Array</td>
<td><code>get_shape()</code></td>
<td>const</td>
</tr>
<tr>
<td>bool</td>
<td><code>is_enabled()</code></td>
<td>const</td>
</tr>
<tr>
<td>void</td>
<td><code>set_connect_range(float range)</code></td>
<td></td>
</tr>
<tr>
<td>void</td>
<td><code>set_disable_distance(float distance)</code></td>
<td></td>
</tr>
<tr>
<td>void</td>
<td><code>set_disabled_color(Color color)</code></td>
<td></td>
</tr>
<tr>
<td>void</td>
<td><code>set_enabled(bool enable)</code></td>
<td></td>
</tr>
<tr>
<td>void</td>
<td><code>set_shape(Vector2Array points)</code></td>
<td></td>
</tr>
</tbody>
</table>

### 12.237.3 Description

Portals provide virtual openings to `VisualInstance` nodes, so cameras can look at them from the outside. Note that portals are a visibility optimization technique, and are in no way related to the game of the same name (as in, they are not used for teleportation). For more information on how rooms and portals work, see `VisualInstance`. Portals are represented as 2D convex polygon shapes (in the X,Y local plane), and are placed on the surface of the areas occupied by a `VisualInstance`, to indicate that the room can be accessed or looked-at through them. If two rooms are next to each other, and two similar portals in each of them share the same world position (and are parallel and opposed to each other), they will automatically “connect” and form “doors” (for example, the portals that connect a kitchen to a living room are placed in the door they share). Portals must always have a `VisualInstance` node as a parent, grandparent or far parent, or else they will not be active.

### 12.237.4 Member Function Description

- **float** `get_connect_range()` *const*
  Return the range for auto-connecting two portals from different rooms sharing the same space.

- **float** `get_disable_distance()` *const*
  Return the distance threshold for disabling the portal. Every time that the portal goes beyond “distance”, it disables itself, becoming the opaque color (see `set_disabled_color`).

- **Color** `get_disabled_color()` *const*
  Return the color for when the portal goes beyond the disable distance (see `set_disable_distance`) and becomes disabled.

- **Vector2Array** `get_shape()` *const*
  Return the portal shape. The shape is an array of `Vector2` points, representing a convex polygon in the X,Y plane.

- **bool** `is_enabled()` *const*
  Return whether the portal is active. When disabled it causes the parent `VisualInstance` to not be visible any longer when looking through the portal.

- **void** `set_connect_range(float range)`
  Set the range for auto-connecting two portals from different rooms sharing the same space.

- **void** `set_disconnect_range(float distance)`
  Set the distance threshold for disabling the portal. Every time that the portal goes beyond “distance”, it disables itself, becoming the opaque color (see `set_disabled_color`).
• void set_disabled_color (Color color)
When the portal goes beyond the disable distance (see set_disable_distance), it becomes opaque and displayed with color “color”.

• void set_enabled (bool enable)
Enable the portal (it is enabled by default though), disabling it will cause the parent VisualInstance to not be visible any longer when looking through the portal.

• void set_shape (Vector2Array points)
Set the portal shape. The shape is an array of Vector2 points, representing a convex polygon in the X,Y plane.

12.238 Position2D

Inherits: Node2D < CanvasItem < Node < Object
Category: Core

12.238.1 Brief Description
Generic 2D Position hint for editing.

12.238.2 Description
Generic 2D Position hint for editing. It’s just like a plain Node2D but displays as a cross in the 2D-Editor at all times.

12.239 Position3D

Inherits: Spatial < Node < Object
Category: Core

12.239.1 Brief Description
Generic 3D Position hint for editing

12.239.2 Description
Generic 3D Position hint for editing. It’s just like a plain Spatial but displays as a cross in the 3D-Editor at all times.

12.240 ProgressBar

Inherits: Range < Control < CanvasItem < Node < Object
Category: Core
12.240.1 Brief Description

General purpose progress bar.

12.240.2 Member Functions

<table>
<thead>
<tr>
<th>bool</th>
<th>is_percent_visible ( ) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>set_percent_visible ( bool visible )</td>
</tr>
</tbody>
</table>

12.240.3 Description

General purpose progress bar. Shows fill percentage from right to left.

12.240.4 Member Function Description

- bool is_percent_visible ( ) const
- void set_percent_visible ( bool visible )

12.241 ProximityGroup

Inherits: Spatial < Node < Object

Category: Core

12.241.1 Brief Description

General purpose proximity-detection node.

12.241.2 Member Functions

<table>
<thead>
<tr>
<th>void</th>
<th>broadcast ( String name, Variant parameters )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vector3</td>
<td>get_grid_radius ( ) const</td>
</tr>
<tr>
<td>void</td>
<td>set_dispatch_mode ( int mode )</td>
</tr>
<tr>
<td>void</td>
<td>set_grid_radius ( Vector3 radius )</td>
</tr>
<tr>
<td>void</td>
<td>set_group_name ( String name )</td>
</tr>
</tbody>
</table>

12.241.3 Signals

- broadcast ( String name, Array parameters )

12.241.4 Description

General purpose proximity-detection node.
12.241.5 Member Function Description

- void broadcast (String name, Variant parameters)
- Vector3 get_grid_radius ( ) const
- void set_dispatch_mode (int mode)
- void set_grid_radius (Vector3 radius)
- void set_group_name (String name)

12.242 Quad

Inherits: GeometryInstance < VisualInstance < Spatial < Node < Object

Category: Core

12.242.1 Brief Description

12.242.2 Member Functions

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>get_axis ( ) const</td>
</tr>
<tr>
<td>Vector2</td>
<td>get_offset ( ) const</td>
</tr>
<tr>
<td>Vector2</td>
<td>get_size ( ) const</td>
</tr>
<tr>
<td>bool</td>
<td>is_centered ( ) const</td>
</tr>
<tr>
<td>void</td>
<td>set_axis (int axis)</td>
</tr>
<tr>
<td>void</td>
<td>set_centered (bool centered)</td>
</tr>
<tr>
<td>void</td>
<td>set_offset (Vector2 offset)</td>
</tr>
<tr>
<td>void</td>
<td>set_size (Vector2 size)</td>
</tr>
</tbody>
</table>

12.242.3 Member Function Description

- int get_axis ( ) const
- Vector2 get_offset ( ) const
- Vector2 get_size ( ) const
- bool is_centered ( ) const
- void set_axis (int axis)
- void set_centered (bool centered)
- void set_offset (Vector2 offset)
- void set_size (Vector2 size)

12.243 Quat

Category: Built-In Types
12.243.1 Brief Description

Quaternion.

12.243.2 Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quat(t float x, float y, float z, float w)</td>
<td>Quaternion constructor.</td>
</tr>
<tr>
<td>Quat(t Vector3 axis, float angle)</td>
<td>Quaternion constructor.</td>
</tr>
<tr>
<td>Quat(t Matrix3 from)</td>
<td>Quaternion constructor.</td>
</tr>
<tr>
<td>quat cubic_slerp Quat b, Quat pre_a, Quat post_b, float t)</td>
<td>Cubic Slerp.</td>
</tr>
<tr>
<td>float dot Quat b</td>
<td>Quaternion dot product.</td>
</tr>
<tr>
<td>Quat inverse</td>
<td>Quaternion inverse.</td>
</tr>
<tr>
<td>float length</td>
<td>Quaternion length.</td>
</tr>
<tr>
<td>float length_squared</td>
<td>Quaternion length squared.</td>
</tr>
<tr>
<td>Quat normalized</td>
<td>Quaternion normalization.</td>
</tr>
<tr>
<td>Quat slerp Quat b, float t</td>
<td>Slerp.</td>
</tr>
<tr>
<td>Quat slerpni Quat b, float t</td>
<td>Slerp with normalization.</td>
</tr>
<tr>
<td>Vector3 xform Quat b, float t</td>
<td>Transform a vector by a quaternion.</td>
</tr>
</tbody>
</table>

12.243.3 Member Variables

- float w
- float x
- float y
- float z

12.243.4 Description

Quaternion is a 4 dimensional vector that is used to represent a rotation. It mainly exists to perform SLERP (spherical-linear interpolation) between rotations obtained by a Matrix3 cheaply. Adding quaternions also cheaply adds the rotations, however quaternions need to be often normalized, or else they suffer from precision issues.

12.243.5 Member Function Description

- Quat(t float x, float y, float z, float w) | Returns the dot product between two quaternions. |
- Quat(t Vector3 axis, float angle) | Returns the inverse of the quaternion (applies to the inverse rotation too). |
- Quat(t Matrix3 from) | Returns the length of the quaternion. |
• `float length_squared()`
  Returns the length of the quaternion, squared.

• `Quat normalized()`
  Returns a copy of the quaternion, normalized to unit length.

• `Quat slerp(Quat b, float t)`
  Perform a spherical-linear interpolation with another quaternion.

• `Quat slerpni(Quat b, float t)`

• `Vector3 xform(Vector3 v)`

### 12.244 Range

Inherits: `Control < CanvasItem < Node < Object`

Inherited By: `SpinBox, ScrollBar, ProgressBar, TextureProgress, Slider`

Category: Core

#### 12.244.1 Brief Description

Abstract base class for range-based controls.

#### 12.244.2 Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>float get_max()</code></td>
<td>const</td>
</tr>
<tr>
<td><code>float get_min()</code></td>
<td>const</td>
</tr>
<tr>
<td><code>float get_page()</code></td>
<td>const</td>
</tr>
<tr>
<td><code>float get_step()</code></td>
<td>const</td>
</tr>
<tr>
<td><code>float get_unit_value()</code></td>
<td>const</td>
</tr>
<tr>
<td><code>float get_value()</code></td>
<td>const</td>
</tr>
<tr>
<td><code>bool is_rounded_values()</code></td>
<td>const</td>
</tr>
<tr>
<td><code>bool is_unit_value_exp()</code></td>
<td>const</td>
</tr>
<tr>
<td><code>void set_exp_unit_value(bool enabled)</code></td>
<td></td>
</tr>
<tr>
<td><code>void set_max(float maximum)</code></td>
<td></td>
</tr>
<tr>
<td><code>void set_min(float minimum)</code></td>
<td></td>
</tr>
<tr>
<td><code>void set_page(float pagesize)</code></td>
<td></td>
</tr>
<tr>
<td><code>void set_rounded_values(bool enabled)</code></td>
<td></td>
</tr>
<tr>
<td><code>void set_step(float step)</code></td>
<td></td>
</tr>
<tr>
<td><code>void set_unit_value(float value)</code></td>
<td></td>
</tr>
<tr>
<td><code>void set_val(float value)</code></td>
<td></td>
</tr>
<tr>
<td><code>void set_value(float value)</code></td>
<td></td>
</tr>
<tr>
<td><code>void share(Object with)</code></td>
<td></td>
</tr>
<tr>
<td><code>void unshare()</code></td>
<td></td>
</tr>
</tbody>
</table>
12.244.3 Signals

- `changed ( )`
  This signal is emitted when min, max, range or step change.
  - `value_changed ( float value )`
  This signal is emitted when value changes.

12.244.4 Description

Range is a base class for Control nodes that change a floating point value between a minimum and a maximum, using step and page, for example a ScrollBar.

12.244.5 Member Function Description

- `float get_max ( ) const`
  Return the maximum value.
- `float get_min ( ) const`
  Return the minimum value.
- `float get_page ( ) const`
  Return the page size, if page is 0, paging is disabled.
- `float get_step ( ) const`
  Return the stepping, if step is 0, stepping is disabled.
- `float get_unit_value ( ) const`
  Return value mapped to 0 to 1 (unit) range.
- `float get_val ( ) const`
  Return the current value.
- `float get_value ( ) const`
- `bool is_rounded_values ( ) const`
- `bool is_unit_value_exp ( ) const`
- `void set_exp_unit_value ( bool enabled )`
- `void set_max ( float maximum )`
- `void set_min ( float minimum )`
  Set minimum value, clamped range value to it if it’s less.
- `void set_page ( float pagesize )`
  Set page size. Page is mainly used for scrollbars or anything that controls text scrolling.
- `void set_rounded_values ( bool enabled )`
- `void set_step ( float step )`
  Set step value. If step is 0, stepping will be disabled.
- `void set_unit_value ( float value )`
Set value mapped to 0 to 1 (unit) range, it will then be converted to the actual value within min and max.

- void set_val (float value)
- void set_value (float value)
- void share (Object with)
- void unshare()

## 12.245 RawArray

**Category:** Built-In Types

### 12.245.1 Brief Description

Raw byte array.

### 12.245.2 Member Functions

<table>
<thead>
<tr>
<th>RawArray</th>
<th>RawArray (Array from)</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>append (int byte)</td>
</tr>
<tr>
<td>void</td>
<td>append_array (RawArray array)</td>
</tr>
<tr>
<td>String</td>
<td>get_string_from_ascii ()</td>
</tr>
<tr>
<td>String</td>
<td>get_string_from_utf8 ()</td>
</tr>
<tr>
<td>int</td>
<td>insert (int idx, int byte)</td>
</tr>
<tr>
<td>void</td>
<td>invert ()</td>
</tr>
<tr>
<td>void</td>
<td>push_back (int byte)</td>
</tr>
<tr>
<td>void</td>
<td>remove (int idx)</td>
</tr>
<tr>
<td>void</td>
<td>resize (int idx)</td>
</tr>
<tr>
<td>void</td>
<td>set (int idx, int byte)</td>
</tr>
<tr>
<td>int</td>
<td>size ()</td>
</tr>
</tbody>
</table>

### 12.245.3 Description

Raw byte array. Contains bytes. Optimized for memory usage, can’t fragment the memory.

### 12.245.4 Member Function Description

- **RawArray RawArray (Array from)**

  Create from a generic array.
  
  - void append (int byte)
  
  - void append_array (RawArray array)

  Append an `RawArray` at the end of this array.
  
  - **String get_string_from_ascii ()**
Returns a copy of the array’s contents formatted as String. Fast alternative to get_string_from_utf8(), assuming the content is ASCII-only (unlike the UTF-8 function, this function maps every byte to a character in the string, so any multibyte sequence will be torn apart).

- **String get_string_from_utf8 ( )**

Returns a copy of the array’s contents formatted as String, assuming the array is formatted as UTF-8. Slower than get_string_from_ascii(), but works for UTF-8. Usually you should prefer this function over get_string_from_ascii() to support international input.

- **int insert ( int idx, int byte )**

Insert a new element at a given position in the array. The position must be valid, or at the end of the array (pos==size()).

- **void invert ( )**

Reverse the order of the elements in the array (so first element will now be the last).

- **void push_back ( int byte )**

Append an element at the end of the array.

- **void remove ( int idx )**

Remove an element from the array by index.

- **void resize ( int idx )**

Set the size of the RawArray. If larger than the current size it will reserve some space beforehand, and if it is smaller it will cut off the array.

- **void set ( int idx, int byte )**

Change the byte at the given index.

- **int size ( )**

Return the size of the array.

## 12.246 RayCast

**Inherits:** Spatial &lt; Node &lt; Object  
**Category:** Core

### 12.246.1 Brief Description

Query the closest object intersecting a ray.
12.246.2 Member Functions

| void | add_exception ( Object node ) |
| void | add_exception_rid ( RID rid ) |
| void | clear_exceptions ( ) |
| void | force-raycast-update ( ) |
| Vector3 | get_cast_to ( ) const |
| Object | get_collider ( ) const |
| int | get_collider_shape ( ) const |
| Vector3 | get_collision_normal ( ) const |
| Vector3 | get_collision_point ( ) const |
| int | get_layer_mask ( ) const |
| int | get_type_mask ( ) const |
| bool | is_colliding ( ) const |
| bool | is_enabled ( ) const |
| void | remove_exception ( Object node ) |
| void | remove_exception_rid ( RID rid ) |
| void | set_cast_to ( Vector3 local_point ) |
| void | set_enabled ( bool enabled ) |
| void | set_layer_mask ( int mask ) |
| void | set_type_mask ( int mask ) |

12.246.3 Description

A RayCast represents a line from its origin to its destination position cast_to, it is used to query the 3D space in order to find the closest object intersecting with the ray.

RayCast can ignore some objects by adding them to the exception list via add_exception, setting proper filtering with layers, or by filtering object types with type masks.

Only enabled raycasts will be able to query the space and report collisions!

RayCast calculates intersection every fixed frame (see Node), and the result is cached so it can be used later until the next frame. If multiple queries are required between fixed frames (or during the same frame) use force-raycast-update after adjusting the raycast.

12.246.4 Member Function Description

- void add_exception ( Object node )

Adds a collision exception so the ray does not report collisions with the specified node.

- void add_exception_rid ( RID rid )
- void clear_exceptions ( )

Removes all collision exception for this ray.

- void force-raycast-update ( )

Updates the collision information in case if this object’s properties changed during the current frame (for example position, rotation or the cast_point). Note, set_enabled is not required for this to work.

- Vector3 get_cast_to ( ) const

Return the destination point of this ray object.
• \texttt{Object get\_collider ( ) const}
Return the closest object the ray is pointing to. Note that this does not consider the length of the vector, so you must also use \texttt{is\_colliding} to check if the object returned is actually colliding with the ray.

• \texttt{int get\_collider\_shape ( ) const}
Returns the collision shape of the closest object the ray is pointing to.

• \texttt{Vector3 get\_collision\_normal ( ) const}
Returns the normal of the intersecting object shape face containing the collision point.

• \texttt{Vector3 get\_collision\_point ( ) const}
Returns collision point. This point is in \texttt{global} coordinate system.

• \texttt{int get\_layer\_mask ( ) const}
Returns the layer mask for this ray.

• \texttt{int get\_type\_mask ( ) const}
Returns the type mask (types of objects to detect) for this ray. The value is a sum (bitwise OR’d) of constants available for \texttt{PhysicsDirectSpaceState}.

• \texttt{bool is\_colliding ( ) const}
Return whether the closest object the ray is pointing to is colliding with the vector (considering the vector length).

• \texttt{bool is\_enabled ( ) const}
Returns whether this raycast is enabled or not.

• \texttt{void remove\_exception ( Object node )}
Removes a collision exception so the ray does report collisions with the specified \texttt{node}.

• \texttt{void remove\_exception\_rid ( RID rid )}
• \texttt{void set\_cast\_to ( Vector3 local\_point )}
Sets to which point ray should be casted. This point is in \texttt{local} coordinate system.

• \texttt{void set\_enabled ( bool enabled )}
Enables the RayCast2D. Only enabled raycasts will be able to query the space and report collisions.

• \texttt{void set\_layer\_mask ( int mask )}
Set the mask to filter objects. Only objects with at least the same mask element set will be detected.

• \texttt{void set\_type\_mask ( int mask )}
Set the types of objects to detect. For \texttt{mask} use a logic sum (OR operation) of constants defined in \texttt{PhysicsDirectSpaceState}, eg. \texttt{PhysicsDirectSpaceState\_TYPE\_MASK\_STATIC\_BODY \textbar\ PhysicsDirectSpaceState\_TYPE\_MASK\_KINEMATIC\_BODY} to detect only those two types.

\section*{12.247 RayCast2D}

\textbf{Inherits:} \texttt{Node2D < CanvasItem < Node < Object}

\textbf{Category:} Core
12.247.1 Brief Description

Query the closest object intersecting a ray.

12.247.2 Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void add_exception (Object node)</td>
<td>Adds a collision exception so the ray does not report collisions with the specified node.</td>
</tr>
<tr>
<td>void add_exception_rid (RID rid)</td>
<td></td>
</tr>
<tr>
<td>void clear_exceptions ( )</td>
<td></td>
</tr>
<tr>
<td>void force_raycast_update ( )</td>
<td></td>
</tr>
<tr>
<td>Vector2 get_cast_to ( ) const</td>
<td></td>
</tr>
<tr>
<td>Object get_collider ( ) const</td>
<td></td>
</tr>
<tr>
<td>int getColliderShape ( ) const</td>
<td></td>
</tr>
<tr>
<td>Vector2 get_collision_normal ( ) const</td>
<td></td>
</tr>
<tr>
<td>Vector2 get_collision_point ( ) const</td>
<td></td>
</tr>
<tr>
<td>int getLayerMask ( ) const</td>
<td></td>
</tr>
<tr>
<td>int getTypeMask ( ) const</td>
<td></td>
</tr>
<tr>
<td>bool isColliding ( ) const</td>
<td></td>
</tr>
<tr>
<td>bool isEnabled ( ) const</td>
<td></td>
</tr>
<tr>
<td>void remove_exception (Object node)</td>
<td></td>
</tr>
<tr>
<td>void remove_exception_rid (RID rid)</td>
<td></td>
</tr>
<tr>
<td>void set_cast_to (Vector2 local_point)</td>
<td></td>
</tr>
<tr>
<td>void set_enabled (bool enabled)</td>
<td></td>
</tr>
<tr>
<td>void set_layer_mask (int mask)</td>
<td></td>
</tr>
<tr>
<td>void set_type_mask (int mask)</td>
<td></td>
</tr>
</tbody>
</table>

12.247.3 Description

A RayCast2D represents a line from its origin to its destination position `cast_to`, it is used to query the 2D space in order to find the closest object intersecting with the ray.

RayCast2D can ignore some objects by adding them to the exception list via `add_exception`, setting proper filtering with layers, or by filtering object types with type masks.

Only enabled raycasts will be able to query the space and report collisions!

RayCast2D calculates intersection every fixed frame (see `Node`), and the result is cached so it can be used later until the next frame. If multiple queries are required between fixed frames (or during the same frame) use `force_raycast_update` after adjusting the raycast.

12.247.4 Member Function Description

- void add_exception (Object node)
  Adds a collision exception so the ray does not report collisions with the specified node.
- void add_exception_rid (RID rid)
- void clear_exceptions ( )
- void force_raycast_update ( )
  Removes all collision exception for this ray.
Updates the collision information in case if this object’s properties changed during the current frame (for example position, rotation or the cast_point). Note, set_enabled is not required for this to work.

• Vector2 get_cast_to() const
Return the destination point of this ray object.

• Object getCollider() const
Return the closest object the ray is pointing to. Note that this does not consider the length of the vector, so you must also use is_colliding to check if the object returned is actually colliding with the ray.

• int getColliderShape() const
Returns the collision shape of the closest object the ray is pointing to.

• Vector2 getCollisionNormal() const
Returns the normal of the intersecting object shape face containing the collision point.

• Vector2 getCollisionPoint() const
Returns the collision point in which the ray intersects the closest object. This point is in global coordinate system.

• int getLayerMask() const
Returns the layer mask for this ray.

• int getTypeMask() const
Returns the type mask (types of objects to detect) for this ray. The value is a sum (bitwise OR’d) of constants available for Physics2DDirectSpaceState.

• bool is_colliding() const
Return whether the closest object the ray is pointing to is colliding with the vector (considering the vector length).

• bool is_enabled() const
Returns whether this raycast is enabled or not.

• void remove_exception(Object node)
Removes a collision exception so the ray does report collisions with the specified node.

• void remove_exception_rid(RID rid)
• void set_cast_to(Vector2 local_point)
Sets the ray destination point, so that the ray will test from the ray’s origin to local_point

• void set_enabled(bool enabled)
Enables the RayCast2D. Only enabled raycasts will be able to query the space and report collisions.

• void set_layer_mask(int mask)
Set the mask to filter objects. Only objects with at least the same mask element set will be detected.

• void set_type_mask(int mask)
Set the types of objects to detect. For mask use a logic sum (OR operation) of constants defined in Physics2DDirectSpaceState, eg. Physics2DDirectSpaceState.TYPE_MASK_STATIC_BODY | Physics2DDirectSpaceState.TYPE_MASK_KINEMATIC_BODY to detect only those two types.
12.248 RayShape

Inherits: Shape < Resource < Reference < Object
Category: Core

12.248.1 Brief Description

12.248.2 Member Functions

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>float</td>
<td>get_length ( ) const</td>
</tr>
<tr>
<td>void</td>
<td>set_length ( float length )</td>
</tr>
</tbody>
</table>

12.248.3 Member Function Description

- float get_length ( ) const
- void set_length ( float length )

12.249 RayShape2D

Inherits: Shape2D < Resource < Reference < Object
Category: Core

12.249.1 Brief Description

Ray 2D shape resource for physics.

12.249.2 Member Functions

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>float</td>
<td>get_length ( ) const</td>
</tr>
<tr>
<td>void</td>
<td>set_length ( float length )</td>
</tr>
</tbody>
</table>

12.249.3 Description

Ray 2D shape resource for physics. A ray is not really a collision body, instead it tries to separate itself from whatever is touching its far endpoint. It’s often useful for characters.

12.249.4 Member Function Description

- float get_length ( ) const
  Return the length of the ray.
- void set_length ( float length )
  Set the length of the ray.
12.250 RealArray

Category: Built-In Types

12.250.1 Brief Description

Real Array.

12.250.2 Member Functions

<table>
<thead>
<tr>
<th>RealArray</th>
<th>RealArray (Array from)</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>append (float value)</td>
</tr>
<tr>
<td>void</td>
<td>append_array (RealArray array)</td>
</tr>
<tr>
<td>int</td>
<td>insert (int idx, float value)</td>
</tr>
<tr>
<td>void</td>
<td>invert ()</td>
</tr>
<tr>
<td>void</td>
<td>push_back (float value)</td>
</tr>
<tr>
<td>void</td>
<td>remove (int idx)</td>
</tr>
<tr>
<td>void</td>
<td>resize (int idx)</td>
</tr>
<tr>
<td>void</td>
<td>set (int idx, float value)</td>
</tr>
<tr>
<td>int</td>
<td>size ()</td>
</tr>
</tbody>
</table>

12.250.3 Description

Real Array. Array of floating point values. Can only contain floats. Optimized for memory usage, can’t fragment the memory.

12.250.4 Member Function Description

- **RealArray RealArray (Array from)**
  Create from a generic array.
  - void append (float value)
    Append an element at the end of the array (alias of push_back).
  - void append_array (RealArray array)
    Append a RealArray at the end of this array.
  - int insert (int idx, float value)
    Insert a new element at a given position in the array. The position must be valid, or at the end of the array (pos==size()).
  - void invert ()
    Reverse the order of the elements in the array (so first element will now be the last).
  - void push_back (float value)
    Append an element at the end of the array.
  - void remove (int idx)
    Remove an element from the array by index.
• void resize ( int idx )

Set the size of the RealArray. If larger than the current size it will reserve some space beforehand, and if it is smaller it will cut off the array.

• void set ( int idx, float value )

Change the float at the given index.

• int size ( )

Return the size of the array.

12.251 Rect2

Category: Built-In Types

12.251.1 Brief Description

2D Axis-aligned bounding box.

12.251.2 Member Functions

| Rect2 | Rect2 ( Vector2 pos, Vector2 size ) |
| Rect2 | Rect2 ( float x, float y, float width, float height ) |
| Rect2 | clip ( Rect2 b ) |
| bool | encloses ( Rect2 b ) |
| Rect2 | expand ( Vector2 to ) |
| float | get_area ( ) |
| Rect2 | grow ( float by ) |
| Rect2 | grow_individual ( float left, float top, float right, float bottom ) |
| Rect2 | grow_margin ( int margin, float by ) |
| bool | has_no_area ( ) |
| bool | has_point ( Vector2 point ) |
| bool | intersects ( Rect2 b ) |
| Rect2 | merge ( Rect2 b ) |

12.251.3 Member Variables

• Vector2 end - Ending corner.

• Vector2 pos - Position (starting corner).

• Vector2 size - Size from position to end.

12.251.4 Description

Rect2 provides an 2D Axis-Aligned Bounding Box. It consists of a position, a size, and several utility functions. It is typically used for fast overlap tests.
### 12.251.5 Member Function Description

- **Rect2 Rect2 ( Vector2 pos, Vector2 size )**
  Construct a Rect2 by position and size.
- **Rect2 Rect2 ( float x, float y, float width, float height )**
  Construct a Rect2 by x, y, width and height.
- **Rect2 clip ( Rect2 b )**
  Returns the intersection of this Rect2 and b.
- **bool encloses ( Rect2 b )**
  Returns true if this Rect2 completely encloses another one.
- **Rect2 expand ( Vector2 to )**
  Return this Rect2 expanded to include a given point.
- **float get_area ( )**
  Get the area of the Rect2.
- **Rect2 grow ( float by )**
  Return a copy of the Rect2 grown a given amount of units towards all the sides.
- **Rect2 grow_individual ( float left, float top, float right, float bottom )**
- **Rect2 grow_margin ( int margin, float by )**
- **bool has_no_area ( )**
  Return true if the Rect2 is flat or empty.
- **bool has_point ( Vector2 point )**
  Return true if the Rect2 contains a point.
- **bool intersects ( Rect2 b )**
  Return true if the Rect2 overlaps with another.
- **Rect2 merge ( Rect2 b )**
  Combine this Rect2 with another, a larger one is returned that contains both.

### 12.252 RectangleShape2D

**Inherits:** Shape2D < Resource < Reference < Object

**Category:** Core

### 12.252.1 Brief Description

Rectangle Shape for 2D Physics.
12.252.2 Member Functions

<table>
<thead>
<tr>
<th>Vector2</th>
<th>get_extents ( ) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>set_extents ( Vector2 extents )</td>
</tr>
</tbody>
</table>

12.252.3 Description

Rectangle Shape for 2D Physics. This shape is useful for modeling box-like 2D objects.

12.252.4 Member Function Description

- Vector2 get_extents ( ) const
  Return the half extents, the actual width and height of this shape is twice the half extents.
- void set_extents ( Vector2 extents )
  Set the half extents, the actual width and height of this shape is twice the half extents.

12.253 Reference

Inherits: Object

Inherited By: RegEx, SurfaceTool, SpatialGizmo, EditorScenePostImport, PhysicsShapeQueryResult, Physics2DTestMotionResult, FuncRef, File, TCP_Server, Physics2DShapeQueryResult, ConfigFile, StreamPeer, HTTPClient, AudioSource, Physics2DShapeQueryParameters, MeshDataTool, GDFunctionState, EditorExportPlugin, EditorScript, Mutex, PacketPeer, Semaphore, XMLParser, EditorImportPlugin, Directory, Marshalls, AStar, SceneState, GDNativeClass, PCKPacker, WeakRef, Resource, Thread, PackedDataContainerRef, ResourceInteractiveLoader, ResourceImportMetadata, PhysicsShapeQueryParameters

Category: Core

12.253.1 Brief Description

Base class for anything that keeps a reference count.

12.253.2 Member Functions

<table>
<thead>
<tr>
<th>bool</th>
<th>init_ref ( )</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>reference ( )</td>
</tr>
<tr>
<td>bool</td>
<td>unrefernce ( )</td>
</tr>
</tbody>
</table>

12.253.3 Description

Base class for anything that keeps a reference count. Resource and many other helper objects inherit this. References keep an internal reference counter so they are only released when no longer in use.
12.253.4 Member Function Description

- bool init_ref ( )
- void reference ( )

Increase the internal reference counter. Use this only if you really know what you are doing.

- bool unreference ( )

Decrease the internal reference counter. Use this only if you really know what you are doing.

12.254 ReferenceFrame

**Inherits:** Control < CanvasItem < Node < Object

**Category:** Core

12.254.1 Brief Description

Reference frame for GUI.

12.254.2 Description

Reference frame for GUI. It’s just like an empty control, except a red box is displayed while editing around its size at all times.

12.255 RegEx

**Inherits:** Reference < Object

**Category:** Core

12.255.1 Brief Description

Simple regular expression matcher.

12.255.2 Member Functions

<table>
<thead>
<tr>
<th>void</th>
<th>clear ( )</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>compile ( String pattern, int capture=9 )</td>
</tr>
<tr>
<td>int</td>
<td>find ( String text, int start=0, int end=-1 ) const</td>
</tr>
<tr>
<td>String</td>
<td>get_capture ( int capture ) const</td>
</tr>
<tr>
<td>int</td>
<td>get_capture_count ( ) const</td>
</tr>
<tr>
<td>int</td>
<td>get_capture_start ( int capture ) const</td>
</tr>
<tr>
<td>StringArray</td>
<td>get_captures ( ) const</td>
</tr>
<tr>
<td>bool</td>
<td>is_valid ( ) const</td>
</tr>
</tbody>
</table>
12.255.3 Description

Class for finding text patterns in a string using regular expressions. It can not perform replacements. Regular expressions are a way to define patterns of text to be searched. Details on writing patterns are too long to explain here but the Internet is full of tutorials and detailed explanations.

Once created, the RegEx object needs to be compiled with the pattern before it can be used. The pattern must be escaped first for gdscript before it is escaped for the expression. For example:

```gdscript
var exp = RegEx.new()
exp.compile("\d+")
```

would be read by RegEx as `\d+`

Similarly:

```gdscript
exp.compile(""(?:\\.|[^\"]\*)")
```

would be read as `"(?:\.\*\")`

Currently supported features:
* Capturing () and non-capturing (?:) groups
* Any character .
* Shorthand character classes \w \W \s \S \d \D
* User-defined character classes such as [A-Za-z]
* Simple quantifiers ?, \* and +
* Range quantifiers (x, y)
* Lazy (non-greedy) quantifiers \*?
* Beginning ^ and end $ anchors
* Alternation |
* Backreferences \1 and \g{1}
* POSIX character classes [[:alnum:]]
* Lookahead (?=), (?!?) and lookbehind (<=), (?<!)
* ASCII \xFF and Unicode \uFFFF code points (in a style similar to Python)
* Word boundaries \b, \B

12.255.4 Member Function Description

- **void clear ( )**
  
  This method resets the state of the object, as it was freshly created. Namely, it unassigns the regular expression of this object, and forgets all captures made by the last find.

- **int compile ( String pattern, int capture=9 )**
  
  Compiles and assign the regular expression pattern to use. The limit on the number of capturing groups can be specified or made unlimited if negative.

- **int find ( String text, int start=0, int end=-1 ) const**
This method tries to find the pattern within the string, and returns the position where it was found. It also stores any capturing group (see get_capture) for further retrieval.

- **String get_capture (int capture) const**
  Returns a captured group. A captured group is the part of a string that matches a part of the pattern delimited by parentheses (unless they are non-capturing parentheses (?)).

- **int get_capture_count () const**
  Returns the number of capturing groups. A captured group is the part of a string that matches a part of the pattern delimited by parentheses (unless they are non-capturing parentheses (?)).

- **IntArray get_capture_start (int capture) const**
  StringArray get_captures () const

  Return a list of all the captures made by the regular expression.

- **bool is_valid () const**
  Returns whether this object has a valid regular expression assigned.

### 12.256 RemoteTransform2D

**Inherits:** Node2D < CanvasItem < Node < Object  
**Category:** Core

#### 12.256.1 Brief Description

#### 12.256.2 Member Functions

<table>
<thead>
<tr>
<th>NodePath</th>
<th>get_remote_node ( ) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>set_remote_node (NodePath path)</td>
</tr>
</tbody>
</table>

#### 12.256.3 Member Function Description

- **NodePath get_remote_node ( ) const**
- **void set_remote_node (NodePath path)**

### 12.257 RenderTargetTexture

**Inherits:** Texture < Resource < Reference < Object  
**Category:** Core

#### 12.257.1 Brief Description

#### 12.258 Resource

**Inherits:** Reference < Object
Godot Engine Documentation, Release 2.1

Inherited By: Theme, AudioManager, EventStream, CubeMap, DynamicFontData, Translation, Curve2D, Shape, Shape2D, Shader, ColorRamp, StyleBox, Material, VideoStream, RoomBounds, PackedScene, Texture, Script, OccluderPolygon2D, Mesh, TileSet, ShortCut, BitMap, Animation, Sample, PolygonPathFinder, BakedLight, World, SampleLibrary, World2D, Font, SpriteFrames, MeshLibrary, Curve3D, NavigationPolygon, EditorSettings, Environment, MultiMesh, CanvasItemMaterial, PackedDataContainer, NavigationMesh

Category: Core

12.258.1 Brief Description

Base class for all resources.

12.258.2 Member Functions

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object duplicate(bool subresources=false)</td>
<td>Duplicate the object with optional subresources.</td>
</tr>
<tr>
<td>Object get_import_metadata() const</td>
<td>Get import metadata of the object.</td>
</tr>
<tr>
<td>String get_name() const</td>
<td>Get the name of the object.</td>
</tr>
<tr>
<td>String get_path() const</td>
<td>Get the path of the object.</td>
</tr>
<tr>
<td>RID get_rid() const</td>
<td>Get the resource ID of the object.</td>
</tr>
<tr>
<td>void set_import_metadata(Object metadata)</td>
<td>Set import metadata.</td>
</tr>
<tr>
<td>void set_name(String name)</td>
<td>Set the name of the object.</td>
</tr>
<tr>
<td>void set_path(String path)</td>
<td>Set the path of the object.</td>
</tr>
<tr>
<td>void take_over_path(String path)</td>
<td>Take over the path.</td>
</tr>
</tbody>
</table>

12.258.3 Signals

- changed()

12.258.4 Description

Resource is the base class for all resource types. Resources are primarily data containers. They are reference counted and freed when no longer in use. They are also loaded only once from disk, and further attempts to load the resource will return the same reference (all this in contrast to a Node, which is not reference counted and can be instantiated from disk as many times as desired). Resources can be saved externally on disk or bundled into another object, such as a Node or another resource.

12.258.5 Member Function Description

- Object duplicate(bool subresources=false)
- Object get_import_metadata() const
- String get_name() const

Return the name of the resources, any name is valid (it doesn’t have to be unique). Name is for descriptive purposes only.

- String get_path() const

Return the path of the resource. This is useful mainly for editors when saving/loading, and shouldn’t be changed by anything else.

- RID get_rid() const
Return the RID of the resource (or an empty RID). Many resources (such as Texture, Mesh, etc) are high level abstractions of resources stored in a server, so this function will return the original RID.

- void set_import_metadata (Object metadata)
- void set_name (String name)

Set the name of the resources, any name is valid (it doesn’t have to be unique). Name is for descriptive purposes only.

- void set_path (String path)

Set the path of the resource. This is useful mainly for editors when saving/loading, and shouldn’t be changed by anything else. Fails if another Resource already has path “path”.

- void take_over_path (String path)

Set the path of the resource. Differs from set_path(), if another Resource exists with “path” it over-takes it, instead of failing.

12.259 ResourceImportMetadata

Inherits: Reference < Object
Category: Core

12.259.1 Brief Description

12.259.2 Member Functions

| void | add_source (String path, String md5="") |
| String | get_editor () const |
| void | get_option (String key) const |
| StringArray | get_options () const |
| int | get_source_count () const |
| String | get_source_md5 (int idx) const |
| String | get_source_path (int idx) const |
| void | remove_source (int idx) |
| void | set_editor (String name) |
| void | set_option (String key, Variant value) |
| void | set_source_md5 (int idx, String md5) |

12.259.3 Member Function Description

- void add_source (String path, String md5="")
- String get_editor () const
- void get_option (String key) const
- StringArray get_options () const
- int get_source_count () const
- String get_source_md5 (int idx) const
- String get_source_path (int idx) const
12.260 ResourceInteractiveLoader

Inherits: Reference < Object

Category: Core

12.260.1 Brief Description

Interactive Resource Loader.

12.260.2 Member Functions

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>get_resource ( )</td>
</tr>
<tr>
<td>int</td>
<td>get_stage ( ) const</td>
</tr>
<tr>
<td>int</td>
<td>get_stage_count ( ) const</td>
</tr>
<tr>
<td>int</td>
<td>poll ( )</td>
</tr>
<tr>
<td>int</td>
<td>wait ( )</td>
</tr>
</tbody>
</table>

12.260.3 Description

Interactive Resource Loader. This object is returned by ResourceLoader when performing an interactive load. It allows to load with high granularity, so this is mainly useful for displaying load bars/percentages.

12.260.4 Member Function Description

- **Object get_resource ( )**
  Return the loaded resource (only if loaded). Otherwise, returns null.

- **int get_stage ( ) const**
  Return the load stage. The total amount of stages can be queried with get_stage_count

- **int get_stage_count ( ) const**
  Return the total amount of stages (calls to poll) needed to completely load this resource.

- **int poll ( )**
  Poll the load. If OK is returned, this means poll will have to be called again. If ERR_FILE_EOF is returned, then the load has finished and the resource can be obtained by calling get_resource.

- **int wait ( )**
12.261 ResourceLoader

Inherits: Object
Category: Core

12.261.1 Brief Description

Resource Loader.

12.261.2 Member Functions

<table>
<thead>
<tr>
<th>Function Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>StringArray get_dependencies</td>
<td>(String path)</td>
</tr>
<tr>
<td>StringArray get_recognized_extensions_for_type</td>
<td>(String type)</td>
</tr>
<tr>
<td>bool has</td>
<td>(String path)</td>
</tr>
<tr>
<td>Resource load</td>
<td>(String path, String type_hint=&quot;&quot;, bool p_no_cache=false)</td>
</tr>
<tr>
<td>ResourceImportMetadata load_import_metadata</td>
<td>(String path)</td>
</tr>
<tr>
<td>ResourceInteractiveLoader load_interactive</td>
<td>(String path, String type_hint=&quot;&quot;)</td>
</tr>
<tr>
<td>void set_abort_on_missing_resources</td>
<td>(bool abort)</td>
</tr>
</tbody>
</table>

12.261.3 Description

Resource Loader. This is a static object accessible as ResourceLoader. GDScript has a simplified load() function, though.

12.261.4 Member Function Description

- **StringArray get_dependencies (String path)**
- **StringArray get_recognized_extensions_for_type (String type)**

Return the list of recognized extensions for a resource type.

- **bool has (String path)**
- **Resource load (String path, String type_hint="", bool p_no_cache=false)**
- **ResourceImportMetadata load_import_metadata (String path)**
- **ResourceInteractiveLoader load_interactive (String path, String type_hint="")**

Load a resource interactively, the returned object allows to load with high granularity.

- **void set_abort_on_missing_resources (bool abort)**

Change the behavior on missing sub-resources. Default is to abort load.

12.262 ResourcePreloader

Inherits: Node < Object
Category: Core
12.262.1 Brief Description

Resource Preloader Node.

12.262.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>add_resource(String name, Object resource)</td>
<td>Add a resource to the preloader. Set the text-id that will be used to identify it (retrieve it/erase it/etc).</td>
</tr>
<tr>
<td>Object</td>
<td>get_resource(String name) const</td>
<td>Return the resource given a text-id.</td>
</tr>
<tr>
<td>StringArray</td>
<td>get_resource_list() const</td>
<td>Return the list of resources inside the preloader.</td>
</tr>
<tr>
<td>bool</td>
<td>has_resource(String name) const</td>
<td>Return true if the preloader has a given resource.</td>
</tr>
<tr>
<td>void</td>
<td>remove_resource(String name)</td>
<td>Remove a resource from the preloader by text id.</td>
</tr>
<tr>
<td>void</td>
<td>rename_resource(String name, String newname)</td>
<td>Rename a resource inside the preloader, from a text-id to a new text-id.</td>
</tr>
</tbody>
</table>

12.262.3 Description

Resource Preloader Node. This node is used to preload sub-resources inside a scene, so when the scene is loaded all the resources are ready to use and be retrieved from here.

12.262.4 Member Function Description

- void add_resource(String name, Object resource)
  Add a resource to the preloader. Set the text-id that will be used to identify it (retrieve it/erase it/etc).
- Object get_resource(String name) const
  Return the resource given a text-id.
- StringArray get_resource_list() const
  Return the list of resources inside the preloader.
- bool has_resource(String name) const
  Return true if the preloader has a given resource.
- void remove_resource(String name)
  Remove a resource from the preloader by text id.
- void rename_resource(String name, String newname)
  Rename a resource inside the preloader, from a text-id to a new text-id.

12.263 ResourceSaver

Inherits: Object
Category: Core

12.263.1 Brief Description

Resource Saving Interface.
12.263.2 Member Functions

<table>
<thead>
<tr>
<th>StringArray</th>
<th>get_recognized_extensions ( Object type )</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>save ( String path, Resource resource, int flags=0 )</td>
</tr>
</tbody>
</table>

12.263.3 Numeric Constants

- FLAG_RELATIVE_PATHS = 1
- FLAG_BUNDLE_RESOURCES = 2
- FLAG_CHANGE_PATH = 4
- FLAG_OMIT_EDITOR_PROPERTIES = 8
- FLAG_SAVE_BIG_ENDIAN = 16
- FLAG_COMPRESS = 32

12.263.4 Description

Resource Saving Interface. This interface is used for saving resources to disk.

12.263.5 Member Function Description

- StringArray get_recognized_extensions ( Object type )
  Return the list of extensions available for saving a resource of a given type.
  - int save ( String path, Resource resource, int flags=0 )
    Save a resource to disk, to a given path.

12.264 RichTextLabel

Inherits: Control < CanvasItem < Node < Object

Category: Core

12.264.1 Brief Description

Label that displays rich text.

12.264.2 Member Functions

<table>
<thead>
<tr>
<th>void</th>
<th>add_image ( Texture image )</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>add_text ( String text )</td>
</tr>
<tr>
<td>int</td>
<td>append_bbcode ( String bbcode )</td>
</tr>
<tr>
<td>void</td>
<td>clear ( )</td>
</tr>
<tr>
<td>String</td>
<td>get_bbcode ( ) const</td>
</tr>
</tbody>
</table>

Continued on next page
Table 12.29 – continued from previous page

<table>
<thead>
<tr>
<th>Function Type</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>float</td>
<td>get_percent_visible</td>
<td>const</td>
</tr>
<tr>
<td>int</td>
<td>get_tab_size</td>
<td>const</td>
</tr>
<tr>
<td>String</td>
<td>get_text</td>
<td></td>
</tr>
<tr>
<td>int</td>
<td>get_total_character_count</td>
<td>const</td>
</tr>
<tr>
<td>VScrollBar</td>
<td>get_v_scroll</td>
<td></td>
</tr>
<tr>
<td>int</td>
<td>get_visible_characters</td>
<td>const</td>
</tr>
<tr>
<td>bool</td>
<td>is_meta_underlined</td>
<td>const</td>
</tr>
<tr>
<td>bool</td>
<td>is_scroll_active</td>
<td>const</td>
</tr>
<tr>
<td>bool</td>
<td>is_scroll_following</td>
<td>const</td>
</tr>
<tr>
<td>bool</td>
<td>is_selection_enabled</td>
<td>const</td>
</tr>
<tr>
<td>bool</td>
<td>is_using_bbcode</td>
<td>const</td>
</tr>
<tr>
<td>void</td>
<td>newline</td>
<td></td>
</tr>
<tr>
<td>int</td>
<td>parse_bbcode</td>
<td>(String bbcode)</td>
</tr>
<tr>
<td>void</td>
<td>pop</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>push_align</td>
<td>(int align)</td>
</tr>
<tr>
<td>void</td>
<td>push_cell</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>push_font</td>
<td>(Font font)</td>
</tr>
<tr>
<td>void</td>
<td>push_color</td>
<td>(Color color)</td>
</tr>
<tr>
<td>void</td>
<td>push_indent</td>
<td>(int level)</td>
</tr>
<tr>
<td>void</td>
<td>push_list</td>
<td>(int type)</td>
</tr>
<tr>
<td>void</td>
<td>push_meta</td>
<td>(Variant data)</td>
</tr>
<tr>
<td>void</td>
<td>push_table</td>
<td>(int columns)</td>
</tr>
<tr>
<td>void</td>
<td>push_underline</td>
<td></td>
</tr>
<tr>
<td>bool</td>
<td>remove_line</td>
<td>(int arg0)</td>
</tr>
<tr>
<td>void</td>
<td>scroll_to_line</td>
<td>(int line)</td>
</tr>
<tr>
<td>void</td>
<td>set_bbcode</td>
<td>(String text)</td>
</tr>
<tr>
<td>void</td>
<td>set_meta_underline</td>
<td>(bool enable)</td>
</tr>
<tr>
<td>void</td>
<td>set_percent_visible</td>
<td>(float percent_visible)</td>
</tr>
<tr>
<td>void</td>
<td>set_scroll_active</td>
<td>(bool active)</td>
</tr>
<tr>
<td>void</td>
<td>set_scroll_follow</td>
<td>(bool follow)</td>
</tr>
<tr>
<td>void</td>
<td>set_selection_enabled</td>
<td>(bool enabled)</td>
</tr>
<tr>
<td>void</td>
<td>set_tab_size</td>
<td>(int spaces)</td>
</tr>
<tr>
<td>void</td>
<td>set_table_column_expand</td>
<td>(int column, bool expand, int ratio)</td>
</tr>
<tr>
<td>void</td>
<td>set_text</td>
<td>(String text)</td>
</tr>
<tr>
<td>void</td>
<td>set_use_bbcode</td>
<td>(bool enable)</td>
</tr>
<tr>
<td>void</td>
<td>set_visible_characters</td>
<td>(int amount)</td>
</tr>
</tbody>
</table>

### 12.264.3 Signals

- meta_clicked (Nil meta)

### 12.264.4 Numeric Constants

- ALIGN_LEFT = 0
- ALIGN_CENTER = 1
- ALIGN_RIGHT = 2
- ALIGN_FILL = 3
- LIST_NUMBERS = 0
• LIST_LETTERS = 1
• LIST_DOTS = 2
• ITEM_FRAME = 0
• ITEM_TEXT = 1
• ITEM_IMAGE = 2
• ITEM_NEWLINE = 3
• ITEM_FONT = 4
• ITEM_COLOR = 5
• ITEM_UNDERLINE = 6
• ITEM_ALIGN = 7
• ITEM_INDENT = 8
• ITEM_LIST = 9
• ITEM_META = 11

12.264.5 Description

Label that displays rich text. Rich text can contain custom text, fonts, images and some basic formatting. It also adapts itself to given width/heights.

12.264.6 Member Function Description

• void add_image (Texture image )
• void add_text (String text )
• int append_bbcode (String bbcode )
• void clear ( )
• String get_bbcode ( ) const
• float get_percent_visible ( ) const
• int get_tab_size ( ) const
• String get_text ( )

Returns the raw text, stripping out the formatting information.

• int get_total_character_count ( ) const
• VScrollBar get_v_scroll ( )
• int get_visible_characters ( ) const
• bool is_meta_underlined ( ) const
• bool is_scroll_active ( ) const
• bool is_scroll_following ( ) const
• bool is_selection_enabled ( ) const

Return true if selecting the text inside this richtext is allowed.
Godot Engine Documentation, Release 2.1

- `bool is_using_bbcode() const`
- `void newline()`
- `int parse_bbcode(String bbcode)`
- `void pop()`
- `void push_align(int align)`
- `void push_cell()`
- `void push_color(Color color)`
- `void push_font(Font font)`
- `void push_indent(int level)`
- `void push_list(int type)`
- `void push_meta(Variant data)`
- `void push_table(int columns)`
- `void push_underline()`
- `bool remove_line(int arg0)`
- `void scroll_to_line(int line)`
- `void set_bbcode(String text)`
- `void set_meta_underline(bool enable)`
- `void set_percent_visible(float percent_visible)`
- `void set_scroll_active(bool active)`
- `void set_scroll_follow(bool follow)`
- `void set_selection_enabled(bool enabled)`

Set to true if selecting the text inside this richtext is allowed.

- `void set_tab_size(int spaces)`
- `void set_table_column_expand(int column, bool expand, int ratio)`
- `void set_text(String text)`
- `void set_use_bbcode(bool enable)`
- `void set_visible_characters(int amount)`

### 12.265 RID

**Category:** Built-In Types

#### 12.265.1 Brief Description

Handle for a `Resource`'s unique ID.
12.265.2 Member Functions

<table>
<thead>
<tr>
<th>RID</th>
<th>RID (Object from)</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>get_id()</td>
</tr>
</tbody>
</table>

12.265.3 Description

The RID type is used to access the unique integer ID of a resource. They are opaque, so they do not grant access to the associated resource by themselves. They are used by and with the low-level Server classes such as VisualServer.

12.265.4 Member Function Description

- **RID RID (Object from)**
  Create a new RID instance with the ID of a given resource. When not handed a valid resource, silently stores the unused ID 0.

- **int get_id()**
  Retrieve the ID of the referenced resource.

12.266 RigidBody

**Inherits:** PhysicsBody < CollisionObject < Spatial < Node < Object

**Category:** Core

12.266.1 Brief Description

Rigid body node.

12.266.2 Member Functions

<table>
<thead>
<tr>
<th>void</th>
<th>_integrate_forces (PhysicsDirectBodyState state) virtual</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>apply_impulse (Vector3 pos, Vector3 impulse)</td>
</tr>
<tr>
<td>float</td>
<td>get_angular_damp() const</td>
</tr>
<tr>
<td>Vector3</td>
<td>get_angular_velocity() const</td>
</tr>
<tr>
<td>int</td>
<td>get_axis_lock() const</td>
</tr>
<tr>
<td>float</td>
<td>get_bounce() const</td>
</tr>
<tr>
<td>Array</td>
<td>get_colliding_bodies() const</td>
</tr>
<tr>
<td>float</td>
<td>get_friction() const</td>
</tr>
<tr>
<td>float</td>
<td>get_gravity_scale() const</td>
</tr>
<tr>
<td>float</td>
<td>get_linear_damp() const</td>
</tr>
<tr>
<td>Vector3</td>
<td>get_linear_velocity() const</td>
</tr>
<tr>
<td>float</td>
<td>get_mass() const</td>
</tr>
<tr>
<td>int</td>
<td>get_max_contacts_reported() const</td>
</tr>
<tr>
<td>int</td>
<td>get_mode() const</td>
</tr>
<tr>
<td>float</td>
<td>get_weight() const</td>
</tr>
</tbody>
</table>

Continued on next page
Table 12.30 – continued from previous page

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool is_able_to_sleep() const</td>
<td></td>
</tr>
<tr>
<td>bool is_contact_monitor_enabled() const</td>
<td></td>
</tr>
<tr>
<td>bool is_sleeping() const</td>
<td></td>
</tr>
<tr>
<td>bool is_using_continuous_collision_detection() const</td>
<td></td>
</tr>
<tr>
<td>bool is_using_custom_integrator()</td>
<td></td>
</tr>
<tr>
<td>void set_angular_damp(float angular_damp)</td>
<td></td>
</tr>
<tr>
<td>void set_angular_velocity(Vector3 angular_velocity)</td>
<td></td>
</tr>
<tr>
<td>void set_axis_lock(int axis_lock)</td>
<td></td>
</tr>
<tr>
<td>void set_axis_velocity(Vector3 axis_velocity)</td>
<td></td>
</tr>
<tr>
<td>void set_bounce(float bounce)</td>
<td></td>
</tr>
<tr>
<td>void set_can_sleep(bool able_to_sleep)</td>
<td></td>
</tr>
<tr>
<td>void set_contact_monitor(bool enabled)</td>
<td></td>
</tr>
<tr>
<td>void set_friction(float friction)</td>
<td></td>
</tr>
<tr>
<td>void set_gravity_scale(float gravity_scale)</td>
<td></td>
</tr>
<tr>
<td>void set_linear_damp(float linear_damp)</td>
<td></td>
</tr>
<tr>
<td>void set_linear_velocity(Vector3 linear_velocity)</td>
<td></td>
</tr>
<tr>
<td>void set_mass(float mass)</td>
<td></td>
</tr>
<tr>
<td>void set_max_contacts_reported(int amount)</td>
<td></td>
</tr>
<tr>
<td>void set_mode(int mode)</td>
<td></td>
</tr>
<tr>
<td>void set_sleepting(bool sleeping)</td>
<td></td>
</tr>
<tr>
<td>void set_use_continuous_collision_detection(bool enable)</td>
<td></td>
</tr>
<tr>
<td>void set_use_custom_integrator(bool enable)</td>
<td></td>
</tr>
<tr>
<td>void set_weight(float weight)</td>
<td></td>
</tr>
</tbody>
</table>

12.266.3 Signals

- **body_enter**(Object body)

  Emitted when a body enters into contact with this one. Contact monitor and contacts reported must be enabled for this to work.

- **body_enter_shape**(int body_id, Object body, int body_shape, int local_shape)

  Emitted when a body enters into contact with this one. Contact monitor and contacts reported must be enabled for this to work. This signal not only receives the body that collided with this one, but also its RID (body_id), the shape index from the colliding body (body_shape), and the shape index from this body (local_shape) the other body collided with.

- **body_exit**(Object body)

  Emitted when a body shape exits contact with this one. Contact monitor and contacts reported must be enabled for this to work.

- **body_exit_shape**(int body_id, Object body, int body_shape, int local_shape)

  Emitted when a body shape exits contact with this one. Contact monitor and contacts reported must be enabled for this to work. This signal not only receives the body that stopped colliding with this one, but also its RID (body_id), the shape index from the colliding body (body_shape), and the shape index from this body (local_shape) the other body stopped colliding with.

- **sleeping_state_changed**( )

  Emitted when the body changes its sleeping state. Either by sleeping or waking up.
12.266.4 Numeric Constants

- **MODE_STATIC = 1** — Static mode. The body behaves like a *StaticBody*, and can only move by user code.
- **MODE_KINEMATIC = 3** — Kinematic body. The body behaves like a *KinematicBody*, and can only move by user code.
- **MODE_RIGID = 0** — Rigid body. This is the “natural” state of a rigid body. It is affected by forces, and can move, rotate, and be affected by user code.
- **MODE_CHARACTER = 2** — Character body. This behaves like a rigid body, but can not rotate.

12.266.5 Description

Rigid body node. This node is used for placing rigid bodies in the scene. It can contain a number of shapes, and also shift mode between regular Rigid body, Kinematic, Character or Static.

12.266.6 Member Function Description

- **void _integrate_forces (PhysicsDirectBodyState state) virtual**

Called during physics processing, allowing you to read and safely modify the simulation state for the object. By default it works in addition to the usual physics behavior, but *set_use_custom_integrator* allows you to disable the default behavior and do fully custom force integration for a body.

- **void apply_impulse (Vector3 pos, Vector3 impulse)**

Apply a positioned impulse (which will be affected by the body mass and shape). This is the equivalent of hitting a billiard ball with a cue: a force that is applied once, and only once. Both the impulse and the offset from the body origin are in global coordinates.

- **float get_angular_damp ( ) const**

Return the current body angular damp. Default is -1.

- **Vector3 get_angular_velocity ( ) const**

Return the current body angular velocity.

- **int get_axis_lock ( ) const**

Return the current axis lock of the body. One of AXIS_LOCK_* enum.

- **float get_bounce ( ) const**

Return the current body bounciness.

- **Array get_colliding_bodies ( ) const**

Return a list of the bodies colliding with this one. By default, number of max contacts reported is at 0, see *set_max_contacts_reported* to increase it.

- **float get_friction ( ) const**

Return the current body friction, from 0 (frictionless) to 1 (max friction).

- **float get_gravity_scale ( ) const**

Return the current body gravity scale.

- **float get_linear_damp ( ) const**

Return the current body linear damp. Default is -1.
- `Vector3 get_linear_velocity ( ) const`
  Return the current body linear velocity.

- `float get_mass ( ) const`
  Return the current body mass.

- `int get_max_contacts_reported ( ) const`
  Return the maximum contacts that can be reported. See `set_max_contacts_reported`.

- `int get_mode ( ) const`
  Return the current body mode, see `set_mode`.

- `float get_weight ( ) const`
  Return the current body weight, given standard earth-weight (gravity 9.8).

- `bool is_able_to_sleep ( ) const`
  Return whether the body has the ability to fall asleep when not moving. See `set_can_sleep`.

- `bool is_contact_monitor_enabled ( ) const`
  Return whether contact monitoring is enabled.

- `bool is_sleeping ( ) const`
  Return whether the body is sleeping.

- `bool is_using_continuous_collision_detection ( ) const`
  Return whether this body is using continuous collision detection.

- `bool is_using_custom_integrator ( )`
  Return whether the body is using a custom integrator.

- `void set_angular_damp ( float angular_damp )`
  Set the angular damp for this body. Default of -1, cannot be less than -1. If this value is different from -1, any angular damp derived from the world or areas will be overridden.

- `void set_angular_velocity ( Vector3 angular_velocity )`
  Set the body angular velocity. Can be used sporadically, but DON'T SET THIS IN EVERY FRAME, because physics may be running in another thread and definitely runs at a different granularity. Use `_integrate_forces` as your process loop if you want to have precise control of the body state.

- `void set_axis_lock ( int axis_lock )`
  Set the axis lock of the body, from the AXIS_LOCK_* enum. Axis lock stops the body from moving along the specified axis(X/Y/Z) and rotating along the other two axes.

- `void set_axis_velocity ( Vector3 axis_velocity )`
  Set an axis velocity. The velocity in the given vector axis will be set as the given vector length. This is useful for jumping behavior.

- `void set_bounce ( float bounce )`
  Set the body bounciness, from 0 (no bounciness) to 1 (max bounciness).

- `void set_can_sleep ( bool able_to_sleep )`
Set the body ability to fall asleep when not moving. This saves an enormous amount of processor time when there are plenty of rigid bodies (non static) in a scene.

Sleeping bodies are not affected by forces until a collision or an `apply_impulse`/`set_applied_force` wakes them up. Until then, they behave like a static body.

- void `set_contact_monitor` ( `bool` enabled )

Enable contact monitoring. This allows the body to emit signals when it collides with another.

- void `set_friction` ( `float` friction )

Set the body friction, from 0 (frictionless) to 1 (max friction).

- void `set_gravity_scale` ( `float` gravity_scale )

Set the gravity factor. This factor multiplies gravity intensity just for this body.

- void `set_linear_damp` ( `float` linear_damp )

Set the linear damp for this body. Default of -1, cannot be less than -1. If this value is different from -1, any linear damp derived from the world or areas will be overridden.

- void `set_linear_velocity` ( `Vector3` linear_velocity )

Set the body linear velocity. Can be used sporadically, but **DON'T SET THIS IN EVERY FRAME**, because physics may be running in another thread and definitely runs at a different granularity. Use `_integrate_forces` as your process loop if you want to have precise control of the body state.

- void `set_mass` ( `float` mass )

Set the body mass.

- void `set_max_contacts_reported` ( `int` amount )

Set the maximum contacts to report. Bodies can keep a log of the contacts with other bodies, this is enabled by setting the maximum amount of contacts reported to a number greater than 0.

- void `set_mode` ( `int` mode )

Set the body mode, from the MODE_* enum. This allows to change to a static body or a character body.

- void `set_sleeeping` ( `bool` sleeping )

Set whether a body is sleeping or not. Sleeping bodies are not affected by forces until a collision or an `apply_impulse` wakes them up. Until then, they behave like a static body.

- void `set_use_continuous_collision_detection` ( `bool` enable )

Set the continuous collision detection mode from the enum CCD_MODE_*.

Continuous collision detection tries to predict where a moving body will collide, instead of moving it and correcting its movement if it collided. The first is more precise, and misses less impacts by small, fast-moving objects. The second is faster to compute, but can miss small, fast-moving objects.

- void `set_use_custom_integrator` ( `bool` enable )

Pass true to disable the internal force integration (like gravity or air friction) for this body. Other than collision response, the body will only move as determined by the `_integrate_forces` function, if defined.

- void `set_weight` ( `float` weight )

Set the body weight given standard earth-weight (gravity 9.8).
12.267 RigidBody2D

Inherits: PhysicsBody2D < CollisionObject2D < Node2D < CanvasItem < Node < Object

Category: Core

12.267.1 Brief Description

Rigid body 2D node.

12.267.2 Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void _integrate_forces (Physics2DDirectBodyState state) virtual</td>
<td></td>
</tr>
<tr>
<td>void add_force (Vector2 offset, Vector2 force)</td>
<td></td>
</tr>
<tr>
<td>void apply_impulse (Vector2 offset, Vector2 impulse)</td>
<td></td>
</tr>
<tr>
<td>float get_angular_damp ( ) const</td>
<td></td>
</tr>
<tr>
<td>float get_angular_velocity ( ) const</td>
<td></td>
</tr>
<tr>
<td>Vector2 get_applied_force ( ) const</td>
<td></td>
</tr>
<tr>
<td>float get_applied_torque ( ) const</td>
<td></td>
</tr>
<tr>
<td>float get_bounce ( ) const</td>
<td></td>
</tr>
<tr>
<td>Array get_colliding_bodies ( ) const</td>
<td></td>
</tr>
<tr>
<td>int get_continuous_collision_detection_mode ( ) const</td>
<td></td>
</tr>
<tr>
<td>float get_friction ( ) const</td>
<td></td>
</tr>
<tr>
<td>float get_gravity_scale ( ) const</td>
<td></td>
</tr>
<tr>
<td>float get_inertia ( ) const</td>
<td></td>
</tr>
<tr>
<td>float get_linear_damp ( ) const</td>
<td></td>
</tr>
<tr>
<td>float get_linear_velocity ( ) const</td>
<td></td>
</tr>
<tr>
<td>float get_mass ( ) const</td>
<td></td>
</tr>
<tr>
<td>int get_max_contacts_reported ( ) const</td>
<td></td>
</tr>
<tr>
<td>int get_mode ( ) const</td>
<td></td>
</tr>
<tr>
<td>float get_weight ( ) const</td>
<td></td>
</tr>
<tr>
<td>bool is_able_to_sleep ( ) const</td>
<td></td>
</tr>
<tr>
<td>bool is_contact_monitor_enabled ( ) const</td>
<td></td>
</tr>
<tr>
<td>bool is_sleeping ( ) const</td>
<td></td>
</tr>
<tr>
<td>bool is_using_custom_integrator ()</td>
<td></td>
</tr>
<tr>
<td>void set_angular_damp (float angular_damp)</td>
<td></td>
</tr>
<tr>
<td>void set_angular_velocity (float angular_velocity)</td>
<td></td>
</tr>
<tr>
<td>void set_applied_force (Vector2 force)</td>
<td></td>
</tr>
<tr>
<td>void set_applied_torque (float torque)</td>
<td></td>
</tr>
<tr>
<td>void set_axis_velocity (Vector2 axis_velocity)</td>
<td></td>
</tr>
<tr>
<td>void set_bounce (float bounce)</td>
<td></td>
</tr>
<tr>
<td>void set_can_sleep (bool able_to_sleep)</td>
<td></td>
</tr>
<tr>
<td>void set_contact_monitor (bool enabled)</td>
<td></td>
</tr>
<tr>
<td>void set_continuous_collision_detection_mode (int mode)</td>
<td></td>
</tr>
<tr>
<td>void set_friction (float friction)</td>
<td></td>
</tr>
<tr>
<td>void set_gravity_scale (float gravity_scale)</td>
<td></td>
</tr>
<tr>
<td>void set_inertia (float inertia)</td>
<td></td>
</tr>
<tr>
<td>void set_linear_damp (float linear_damp)</td>
<td></td>
</tr>
<tr>
<td>void set_linear_velocity (Vector2 linear_velocity)</td>
<td></td>
</tr>
</tbody>
</table>
12.267.3 Signals

- **body_enter** *(Object body)*

  Emitted when a body enters into contact with this one. Contact monitor and contacts reported must be enabled for this to work.

- **body_enter_shape** *(int body_id, Object body, int body_shape, int local_shape)*

  Emitted when a body enters into contact with this one. Contact monitor and contacts reported must be enabled for this to work.

  This signal not only receives the body that collided with this one, but also its RID (body_id), the shape index from the colliding body (body_shape), and the shape index from this body (local_shape) the other body collided with.

- **body_exit** *(Object body)*

  Emitted when a body exits contact with this one. Contact monitor and contacts reported must be enabled for this to work.

- **body_exit_shape** *(int body_id, Object body, int body_shape, int local_shape)*

  Emitted when a body shape exits contact with this one. Contact monitor and contacts reported must be enabled for this to work.

  This signal not only receives the body that stopped colliding with this one, but also its RID (body_id), the shape index from the colliding body (body_shape), and the shape index from this body (local_shape) the other body stopped colliding with.

- **sleeping_state_changed** *( )*

  Emitted when the body changes its sleeping state. Either by sleeping or waking up.

12.267.4 Numeric Constants

- **MODE_STATIC = 1** — Static mode. The body behaves like a StaticBody2D, and can only move by user code.

- **MODE_KINEMATIC = 3** — Kinematic body. The body behaves like a KinematicBody2D, and can only move by user code.

- **MODE_RIGID = 0** — Rigid body. This is the “natural” state of a rigid body. It is affected by forces, and can move, rotate, and be affected by user code.

- **MODE_CHARACTER = 2** — Character body. This behaves like a rigid body, but can not rotate.

- **CCD_MODE_DISABLED = 0** — Disables continuous collision detection. This is the fastest way to detect body collisions, but can miss small, fast-moving objects.

- **CCD_MODE_CAST_RAY = 1** — Enables continuous collision detection by raycasting. It is faster than shapcasting, but less precise.
- **CCD_MODE_CAST_SHAPE = 2** — Enables continuous collision detection by shapecasting. It is the slowest CCD method, and the most precise.

### 12.267.5 Description

Rigid body 2D node. This node is used for placing rigid bodies in the scene. It can contain a number of shapes, and also shift state between regular Rigid body, Kinematic, Character or Static.

Character mode forbids the node from being rotated. This node can have a custom force integrator function, for writing complex physics motion behavior per node.

As a warning, don’t change this node position every frame or very often. Sporadic changes work fine, but physics runs at a different granularity (fixed hz) than usual rendering (process callback) and maybe even in a separate thread, so changing this from a process loop will yield strange behavior.

### 12.267.6 Member Function Description

- **void _integrate_forces (Physics2DDirectBodyState state) virtual**
  
  Called during physics processing, allowing you to read and safely modify the simulation state for the object. By default it works in addition to the usual physics behavior, but `set_use_custom_integrator` allows you to disable the default behavior and do fully custom force integration for a body.

- **void add_force (Vector2 offset, Vector2 force)**
  
  Add a positioned force to the applied force and torque. As with `apply_impulse`, both the force and the offset from the body origin are in global coordinates.

- **void apply_impulse (Vector2 offset, Vector2 impulse)**
  
  Apply a positioned impulse (which will be affected by the body mass and shape). This is the equivalent of hitting a billiard ball with a cue: a force that is applied once, and only once. Both the impulse and the offset from the body origin are in global coordinates.

- **float get_angular_damp () const**
  
  Return the angular damp for this body.

- **float get_angular_velocity () const**
  
  Return the body angular velocity. This changes by physics granularity. See `set_angular_velocity`.

- **Vector2 get_applied_force () const**
  
  Return the applied force vector.

- **float get_applied_torque () const**
  
  Return the torque which is being applied to this body.

- **float get_bounce () const**
  
  Return the body bounciness.

- **Array get_colliding_bodies () const**
  
  Return a list of the bodies colliding with this one. By default, number of max contacts reported is at 0, see `set_max_contacts_reported` to increase it. You must also enable contact monitor, see `set_contact_monitor`.

- **int get_continuous_collision_detection_mode () const**
  
  Return whether this body is using continuous collision detection.
- `float get_friction ( ) const`
  Return the body friction.

- `float get_gravity_scale ( ) const`
  Return the gravity factor.

- `float get_inertia ( ) const`
  Return the body’s moment of inertia. This is usually automatically computed from the mass and the shapes. Note that this doesn’t seem to work in a `_ready` function: it apparently has not been auto-computed yet.

- `float get_linear_damp ( ) const`
  Return the linear damp for this body.

- `Vector2 get_linear_velocity ( ) const`
  Return the body linear velocity. This changes by physics granularity. See `set_linear_velocity`.

- `float get_mass ( ) const`
  Return the body mass.

- `int get_max_contacts_reported ( ) const`
  Return the maximum contacts that can be reported. See `set_max_contacts_reported`.

- `int get_mode ( ) const`
  Return the current body mode, see `set_mode`.

- `float get_weight ( ) const`
  Return the body weight given standard earth-weight (gravity 9.8).

- `bool is_able_to_sleep ( ) const`
  Return true if the body has the ability to fall asleep when not moving. See `set_can_sleep`.

- `bool is_contact_monitor_enabled ( ) const`
  Return whether contact monitoring is enabled.

- `bool is_sleeping ( ) const`
  Return whether the body is sleeping.

- `bool is_using_custom_integrator ( )`
  Return true if the body is not doing any built-in force integration.

- `void set_angular_damp ( float angular_damp )`
  Set the angular damp for this body. If this value is different from -1, any angular damp derived from the world or areas will be overridden.

- `void set_angular_velocity ( float angular_velocity )`
  Set the body angular velocity. Can be used sporadically, but DON’T SET THIS IN EVERY FRAME, because physics may be running in another thread and definitely runs at a different granularity. Use `_integrate_forces` as your process loop if you want to have precise control of the body state.

- `void set_applied_force ( Vector2 force )`
  Set the applied force vector. This is the equivalent of pushing a box over the ground: the force applied is applied constantly.

- `void set_applied_torque ( float torque )`
Set a constant torque which will be applied to this body.

- void **set_axis_velocity** ( Vector2 axis_velocity )

Set an axis velocity. The velocity in the given vector axis will be set as the given vector length. This is useful for jumping behavior.

- void **set_bounce** ( float bounce )

Set the body bounciness, from 0 (no bounce) to 1 (full bounce).

- void **set_can_sleep** ( bool able_to_sleep )

Set the body ability to fall asleep when not moving. This saves an enormous amount of processor time when there are plenty of rigid bodies (non static) in a scene.

Sleeping bodies are not affected by forces until a collision or an apply_impulse / set_applied_force wakes them up. Until then, they behave like a static body.

- void **set_contact_monitor** ( bool enabled )

Enable contact monitoring. This allows the body to emit signals when it collides with another.

- void **set_continuous_collision_detection_mode** ( int mode )

Set the continuous collision detection mode from the enum CCD_MODE_.*.

Continuous collision detection tries to predict where a moving body will collide, instead of moving it and correcting its movement if it collided. The first is more precise, and misses less impacts by small, fast-moving objects. The second is faster to compute, but can miss small, fast-moving objects.

- void **set_friction** ( float friction )

Set the body friction, from 0 (frictionless) to 1 (full friction).

- void **set_gravity_scale** ( float gravity_scale )

Set the gravity factor. This factor multiplies gravity intensity just for this body.

- void **set_inertia** ( float inertia )

Set the body’s moment of inertia. This is like mass, but for rotation: it determines how much torque it takes to rotate the body. The moment of inertia is usually computed automatically from the mass and the shapes, but this function allows you to set a custom value. Set 0 (or negative) inertia to return to automatically computing it.

- void **set_linear_damp** ( float linear_damp )

Set the linear damp for this body. If this value is different from -1, any linear damp derived from the world or areas will be overridden.

- void **set_linear_velocity** ( Vector2 linear_velocity )

Set the body linear velocity. Can be used sporadically, but DON’T SET THIS IN EVERY FRAME, because physics may be running in another thread and definitely runs at a different granularity. Use _integrate_forces as your process loop if you want to have precise control of the body state.

- void **set_mass** ( float mass )

Set the body mass.

- void **set_max_contacts_reported** ( int amount )

Set the maximum contacts to report. Bodies can keep a log of the contacts with other bodies, this is enabled by setting the maximum amount of contacts reported to a number greater than 0.

- void **set_mode** ( int mode )

Set the body mode, from the MODE_.* enum. This allows to change to a static body or a character body.
• void set_sleeping ( bool sleeping )
Set whether a body is sleeping or not. Sleeping bodies are not affected by forces until a collision or an apply_impulse / set_applied_force wakes them up. Until then, they behave like a static body.

• void set_use_custom_integrator ( bool enable )
Pass true to disable the internal force integration (like gravity or air friction) for this body. Other than collision response, the body will only move as determined by the _integrate_forces function, if defined.

• void set_weight ( float weight )
Set the body weight given standard earth-weight (gravity 9.8). Not really useful for 2D since most measures for this node are in pixels.

• bool test_motion ( Vector2 motion, float margin=0.08, Physics2DTestMotionResult result=NULL )
Return whether the body would collide, if it tried to move in the given vector. This method allows two extra parameters: A margin, which increases slightly the size of the shapes involved in the collision detection, and an object of type Physics2DTestMotionResult, which will store additional information about the collision (should there be one).

12.268 Room

Inherits: VisualInstance < Spatial < Node < Object
Category: Core

12.268.1 Brief Description
Room data resource.

12.268.2 Member Functions

<table>
<thead>
<tr>
<th>void</th>
<th>compute_room_from_subtree ( )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Room</td>
<td>get_room ( ) const</td>
</tr>
<tr>
<td>bool</td>
<td>is_simulating_acoustics ( ) const</td>
</tr>
<tr>
<td>void</td>
<td>set_room ( Room room )</td>
</tr>
<tr>
<td>void</td>
<td>set_simulate_acoustics ( bool enable )</td>
</tr>
</tbody>
</table>

12.268.3 Description
Room contains the data to define the bounds of a scene (using a BSP Tree). It is instanced by a VisualInstance node to create rooms. See that class documentation for more information about rooms.

12.268.4 Member Function Description

• void compute_room_from_subtree ( )
• Room get_room ( ) const
• bool is_simulating_acoustics ( ) const
• void set_room ( Room room )
• void set_simulate_acoustics ( bool enable )
12.269 RoomBounds

Inherits: Resource \(<\) Reference \(<\) Object

Category: Core

12.269.1 Brief Description

12.269.2 Member Functions

<table>
<thead>
<tr>
<th>Dictionary</th>
<th>get_bounds ( ) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vector3Array</td>
<td>get_geometry_hint ( ) const</td>
</tr>
<tr>
<td>void</td>
<td>regenerate_bsp ( )</td>
</tr>
<tr>
<td>void</td>
<td>regenerate_bsp_cubic ( )</td>
</tr>
<tr>
<td>void</td>
<td>set_bounds ( Dictionary bsp_tree )</td>
</tr>
<tr>
<td>void</td>
<td>set_geometry_hint ( Vector3Array triangles )</td>
</tr>
</tbody>
</table>

12.269.3 Member Function Description

- Dictionary get_bounds ( ) const
- Vector3Array get_geometry_hint ( ) const
- void regenerate_bsp ( )
- void regenerate_bsp_cubic ( )
- void set_bounds ( Dictionary bsp_tree )
- void set_geometry_hint ( Vector3Array triangles )

12.270 Sample

Inherits: Resource \(<\) Reference \(<\) Object

Category: Core

12.270.1 Brief Description

Audio sample (sound) class.
### 12.270.2 Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>create(int format, bool stereo, int length)</code></td>
<td>Create new data for the sample, with format (see FORMAT_* constants), stereo hint, and length in samples (not bytes). Calling this method overrides previously existing data. Stereo samples are interleaved pairs of left and right points (in that order), but count as one sample for length purposes.</td>
</tr>
<tr>
<td><code>get_data()</code> const</td>
<td>Return sample data as little endian.</td>
</tr>
<tr>
<td><code>get_format()</code> const</td>
<td>Return the sample format.</td>
</tr>
<tr>
<td><code>get_length()</code> const</td>
<td>Return the sample length in samples. Stereo samples count as one, even if they are made of a left and a right sample.</td>
</tr>
</tbody>
</table>

### 12.270.3 Numeric Constants

- **FORMAT_PCM8 = 0** — 8-bits signed PCM audio.
- **FORMAT_PCM16 = 1** — 16-bits signed little endian PCM audio.
- **FORMAT_ILM_ADPCM = 2** — IMA-ADPCM Audio.
- **LOOP_NONE = 0** — No loop enabled.
- **LOOP_FORWARD = 1** — Forward looping (when playback reaches loop end, goes back to loop begin).
- **LOOP_PING_PONG = 2** — Ping-pong looping (when playback reaches loop end, plays backward until loop begin). Not available in all platforms.

### 12.270.4 Description

Sample provides an audio sample class, containing audio data, together with some information for playback, such as format, mix rate and loop. It is used by sound playback routines.

### 12.270.5 Member Function Description

- **void create(int format, bool stereo, int length)**
  Create new data for the sample, with format (see FORMAT_* constants), stereo hint, and length in samples (not bytes). Calling this method overrides previously existing data. Stereo samples are interleaved pairs of left and right points (in that order), but count as one sample for length purposes.

- **RawArray get_data() const**
  Return sample data as little endian.

- **int get_format() const**
  Return the sample format.

- **int get_length() const**
  Return the sample length in samples. Stereo samples count as one, even if they are made of a left and a right sample.
- `int get_loop_begin()` const
  Return the loop begin position.

- `int get_loop_end()` const
  Return the loop end position.

- `int get_loop_format()` const
  Return the loop format.

- `int get_mix_rate()` const
  Return the mix rate for the sample.

- `bool is_stereo()` const
  Return whether the current sample was created as stereo.

- `void set_data(RawArray data)`
  Set sample data. Data must be little endian, no matter the host platform, and exactly as long as to fit all samples. The length of this array can be calculated as follows:

  Get the sample length (`get_length`). If the sample format is FORMAT_PCM16, multiply it by 2. If the sample format is FORMAT.IMA_ADPCM, divide it by 2 (rounding any fraction up), then add 4. If the sample is stereo (`is_stereo`), multiply it by 2.

- `void set_loop_begin(int pos)`
  Set the loop begin position. It must be a valid frame and less than the loop end position.

- `void set_loop_end(int pos)`
  Set the loop end position. It must be a valid frame and greater than the loop begin position.

- `void set_loop_format(int format)`
  Set the loop format (use LOOP_* constants as argument).

- `void set_mix_rate(int hz)`
  Set the mix rate for the sample (expected playback frequency).

### 12.271 SampleLibrary

Inherits: `Resource < Reference < Object`

Category: Core

#### 12.271.1 Brief Description

Library that contains a collection of samples.
12.271.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>add_sample (String name, Sample sample)</td>
</tr>
<tr>
<td>Sample</td>
<td>get_sample (String name) const</td>
</tr>
<tr>
<td>Array</td>
<td>get_sample_list () const</td>
</tr>
<tr>
<td>bool</td>
<td>has_sample (String name) const</td>
</tr>
<tr>
<td>void</td>
<td>remove_sample (String name)</td>
</tr>
<tr>
<td>float</td>
<td>sample_get_pitch_scale (String name) const</td>
</tr>
<tr>
<td>int</td>
<td>sample_get_priority (String name) const</td>
</tr>
<tr>
<td>float</td>
<td>sample_get_volume_db (String name) const</td>
</tr>
<tr>
<td>void</td>
<td>sample_set_pitch_scale (String name, float pitch)</td>
</tr>
<tr>
<td>void</td>
<td>sample_set_priority (String name, int priority)</td>
</tr>
<tr>
<td>void</td>
<td>sample_set_volume_db (String name, float db)</td>
</tr>
</tbody>
</table>

12.271.3 Description

Library that contains a collection of Sample, each identified by a text ID. This is used as a data container for the majority of the SamplePlayer classes and derivatives.

Sample players will never yield an active (currently playing) voice for a new playback request when there are no inactive voices available if the priority of the sample requested to be played is lower than that of every currently played samples.

12.271.4 Member Function Description

- **void add_sample (String name, Sample sample)**
  Add a sample to the library, with a given text ID.

- **Sample get_sample (String name) const**
  Return the sample from the library matching the given text ID. Return null if the sample is not found.

- **Array get_sample_list () const**
- **bool has_sample (String name) const**
  Return true if the sample text ID exists in the library.

- **void remove_sample (String name)**
  Remove the sample matching the given text ID.

- **float sample_get_pitch_scale (String name) const**
  Return the pitch scale for the given sample.

- **int sample_get_priority (String name) const**
  Return the priority for the given sample.

- **float sample_get_volume_db (String name) const**
  Return the volume (in dB) for the given sample.

- **void sample_set_pitch_scale (String name, float pitch)**
  Set the pitch scale for the given sample.

- **void sample_set_priority (String name, int priority)**
Set the priority for the given sample.

- void `sample_set_volume_db(String name, float db)`

Set the volume (in dB) for the given sample.

### 12.272 SamplePlayer

**Inherits:** `Node < Object`

**Category:** Core

#### 12.272.1 Brief Description

Sample Player node.

#### 12.272.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>float</code></td>
<td><code>get_chorus(int voice)</code> const</td>
</tr>
<tr>
<td><code>float</code></td>
<td><code>get_default_chorus()</code> const</td>
</tr>
<tr>
<td><code>float</code></td>
<td><code>get_default_filter_cutoff()</code> const</td>
</tr>
<tr>
<td><code>float</code></td>
<td><code>get_default_filter_gain()</code> const</td>
</tr>
<tr>
<td><code>float</code></td>
<td><code>get_default_filter_resonance()</code> const</td>
</tr>
<tr>
<td><code>int</code></td>
<td><code>get_default_filter_type()</code> const</td>
</tr>
<tr>
<td><code>float</code></td>
<td><code>get_default_pan()</code> const</td>
</tr>
<tr>
<td><code>float</code></td>
<td><code>get_default_pan_depth()</code> const</td>
</tr>
<tr>
<td><code>float</code></td>
<td><code>get_default_pan_height()</code> const</td>
</tr>
<tr>
<td><code>float</code></td>
<td><code>get_default_pitch_scale()</code> const</td>
</tr>
<tr>
<td><code>float</code></td>
<td><code>get_default_reverb()</code> const</td>
</tr>
<tr>
<td><code>int</code></td>
<td><code>get_default_reverb_room()</code> const</td>
</tr>
<tr>
<td><code>float</code></td>
<td><code>get_default_volume()</code> const</td>
</tr>
<tr>
<td><code>float</code></td>
<td><code>get_default_volume_db()</code> const</td>
</tr>
<tr>
<td><code>float</code></td>
<td><code>get_filter_cutoff(int voice)</code> const</td>
</tr>
<tr>
<td><code>float</code></td>
<td><code>get_filter_gain(int voice)</code> const</td>
</tr>
<tr>
<td><code>float</code></td>
<td><code>get_filter_resonance(int voice)</code> const</td>
</tr>
<tr>
<td><code>int</code></td>
<td><code>get_filter_type(int voice)</code> const</td>
</tr>
<tr>
<td><code>int</code></td>
<td><code>get_mix_rate(int voice)</code> const</td>
</tr>
<tr>
<td><code>float</code></td>
<td><code>get_pan(int voice)</code> const</td>
</tr>
<tr>
<td><code>float</code></td>
<td><code>get_pan_depth(int voice)</code> const</td>
</tr>
<tr>
<td><code>float</code></td>
<td><code>get_pan_height(int voice)</code> const</td>
</tr>
<tr>
<td><code>float</code></td>
<td><code>get_pitch_scale(int voice)</code> const</td>
</tr>
<tr>
<td><code>int</code></td>
<td><code>get_polyphony()</code> const</td>
</tr>
<tr>
<td><code>float</code></td>
<td><code>get_reverb(int voice)</code> const</td>
</tr>
<tr>
<td><code>int</code></td>
<td><code>get_reverb_room(int voice)</code> const</td>
</tr>
<tr>
<td><code>SampleLibrary</code></td>
<td><code>get_sample_library()</code> const</td>
</tr>
<tr>
<td><code>float</code></td>
<td><code>get_volume(int voice)</code> const</td>
</tr>
<tr>
<td><code>float</code></td>
<td><code>get_volume_db(int voice)</code> const</td>
</tr>
<tr>
<td><code>bool</code></td>
<td><code>is_active()</code> const</td>
</tr>
<tr>
<td><code>bool</code></td>
<td><code>is_voice_active(int voice)</code> const</td>
</tr>
</tbody>
</table>

Continued on next page
### 12.272.3 Numeric Constants

- **FILTER_NONE** = 0 — Filter is disabled for voice.
- **FILTER_LOWPASS** = 1 — Low-pass filter is used for voice.
- **FILTER_BANDPASS** = 2 — Band-pass filter is used for voice.
- **FILTER_HIPASS** = 3 — High-pass filter is used for voice.
- **FILTER_NOTCH** = 4 — Notch (band reject) filter is used for voice.
- **FILTER_PEAK** = 5 — Peak (exclusive band) filter is used for voice.
- **FILTER_BANDLIMIT** = 6 — Band-limit filter is used for voice, in this case resonance is the high-pass cutoff. A band-limit filter has a different frequency response than a notch filter, but otherwise both are band-rejecting filters.
- **FILTER_LOW_SHELF** = 7 — Low-shelf filter is used for voice.
- **FILTER_HIGH_SHELF** = 8 — High-shelf filter is used for voice.
- **REVERB_SMALL** = 0 — Small reverberation room (house room).
- **REVERB_MEDIUM** = 1 — Medium reverberation room (street)
- **REVERB_LARGE** = 2 — Large reverberation room (theatre)
- **REVERB_HALL** = 3 — Huge reverberation room (cathedral, warehouse).
- **INVALID_VOICE_ID** = -1 — Value returned if the voice ID is invalid.

### 12.272.4 Description

SamplePlayer is a Node meant for simple sample playback. A library of samples is loaded and played back “as is”, without positioning or anything.
12.272.5 Member Function Description

- float get_chorus (int voice) const
  Return the current chorus send level for a given voice.

- float get_default_chorus ( ) const
  Return the default chorus send level of the player.

- float get_default_filter_cutoff ( ) const
  Return the default filter cutoff frequency of the player.

- float get_default_filter_gain ( ) const
  Return the default filter gain of the player.

- float get_default_filter_resonance ( ) const
  Return the default filter resonance of the player.

- int get_default_filter_type ( ) const
  Return the default filter type in use (see FILTER_* constants) for the player.

- float get_default_pan ( ) const
  Return the default panning of the player.

- float get_default_pan_depth ( ) const
  Return the default pan depth of the player.

- float get_default_pan_height ( ) const
  Return the default pan height of the player.

- float get_default_pitch_scale ( ) const
  Return the default pitch scale of the player.

- float get_default_reverb ( ) const
  Return the default reverberation send level of the player.

- int get_default_reverb_room ( ) const
  Return the default reverberation room type of the player (see REVERB_* enum).

- float get_default_volume ( ) const
  Return the default volume (on a linear scale) of the player.

- float get_default_volume_db ( ) const
  Return the default volume (in dB) of the player.

- float get_filter_cutoff (int voice) const
  Return the current filter cutoff frequency for a given voice.

- float get_filter_gain (int voice) const
  Return the current filter gain for a given voice.

- float get_filter_resonance (int voice) const
  Return the current filter resonance for a given voice.

- int get_filter_type (int voice) const
Return the current filter type in use (see FILTER_* constants) for a given voice.

• `int get_mix_rate (int voice) const`

Return the current mix rate for a given voice.

• `float get_pan (int voice) const`

Return the current panning for a given voice.

• `float get_pan_depth (int voice) const`

Return the current pan depth for a given voice.

• `float get_pan_height (int voice) const`

Return the current pan height for a given voice.

• `float get_pitch_scale (int voice) const`

Return the current pitch scale for a given voice.

• `int get_polyphony () const`

Return the polyphony of the player.

• `float get_reverb (int voice) const`

Return the current reverberation send level for a given voice.

• `int get_reverb_room (int voice) const`

Return the current reverberation room type for a given voice (see REVERB_* enum).

• `SampleLibrary get_sample_library () const`

Return the sample library used by the player.

• `float get_volume (int voice) const`

Return the current volume (on a linear scale) for a given voice.

• `float get_volume_db (int voice) const`

Return the current volume (in dB) for a given voice.

• `bool is_active () const`

Return whether the player is currently active.

• `bool is_voice_active (int voice) const`

Return whether the given voice is currently active.

• `int play (String name, bool unique=false)`

Play a sample referenced by its name.

Optionally, the playback can be made “unique” to force stopping all other samples currently played. The voices allocated for playback will then be returned.

• `void set_chorus (int voice, float send)`

Set the chorus send level of a voice (from 0 to 1.0). For setting chorus parameters, see AudioServer.

• `void set_default_chorus (float send)`

Set the default chorus send level of the player (from 0 to 1.0). For setting chorus parameters, see AudioServer.

• `void set_default_filter (int type, float cutoff_hz, float resonance, float gain=0)`
Set the default filter for the player, using the given type (see FILTER_* constants), cutoff frequency (from 20 to 16,384 Hz) and resonance (from 0 to 4.0).

Optionally, a gain can also be given (from 0 to 2.0).

- void set_default_pan ( float pan, float depth=0, float height=0 )
  Set the default panning of the player. Panning goes from -1.0 (left) to +1.0 (right).

Optionally, for hardware than support 3D sound, one can also set depth and height (also in range -1.0 to +1.0).

- void set_default_pitch_scale ( float ratio )
  Set the default pitch scale of the player. A ratio of 1.0 is the normal scale.

- void set_default_reverb ( int room_type, float send )
  Set the default reverberation type (see REVERB_* constants) and send level (from 0 to 1.0) of the player.

- void set_default_volume ( float volume )
  Set the default volume of the player using a linear scale.
  The “volume” argument should be a positive factor ranging from 0.0 (mute) up to 16.0 (i.e. 24 dB).
  A factor of 1.0 means that the voice will be played at normal system volume. Factors above 1.0 might be limited by the platform’s audio output.

- void set_default_volume_db ( float db )
  Set the default volume of the player in dB.
  The “dB” argument can range from -80 to 24 dB, 0 dB being the maximum volume. Every 6 dB (resp. -6 dB), the volume is increased (resp. reduced) by half.

- void set_filter ( int voice, int type, float cutoff_hz, float resonance, float gain=0 )
  Set the filter for a given voice, using the given type (see FILTER_* constants), cutoff frequency (from 20 to 16,384 Hz) and resonance (from 0 to 4.0).
  Optionally, a gain can also be given (from 0 to 2.0).

- void set_mix_rate ( int voice, int hz )
  Set the mix rate (in Hz) of a given voice.

- void set_pan ( int voice, float pan, float depth=0, float height=0 )
  Set the panning of a voice. Panning goes from -1.0 (left) to +1.0 (right).

Optionally, for hardware than support 3D sound, one can also set depth and height (also in range -1.0 to +1.0).

- void set_pitch_scale ( int voice, float ratio )
  Set the pitch scale of a given voice. A ratio of 1.0 is the normal scale.

- void set_polyphony ( int max_voices )
  Set the polyphony of the player (maximum amount of simultaneous voices).

- void set_reverb ( int voice, int room_type, float send )
  Set the reverberation type (see REVERB_* constants) and send level (from 0 to 1.0) of a voice.

- void set_sample_library ( SampleLibrary library )
  Set the sample library for the player.

- void set_volume ( int voice, float volume )
  Set the volume of a given voice.
Set the volume of a given voice using a linear scale.
The “volume” argument should be a positive factor ranging from 0.0 (mute) up to 16.0 (i.e. 24 dB).
A factor of 1.0 means that the voice will be played at normal system volume. Factors above 1.0 might be limited by
the platform’s audio output.

- void set_volume_db ( int voice, float db )

Set the volume of a given voice in dB.
The “dB” argument can range from -80 to 24 dB, 0 dB being the maximum volume. Every 6 dB (resp. -6 dB), the
volume is increased (resp. reduced) by half.

- void stop ( int voice )

Stop a given voice.

- void stop_all ( )

Stop all playing voices.

12.273 SamplePlayer2D

Inherits: SoundPlayer2D < Node2D < CanvasItem < Node < Object

Category: Core

12.273.1 Brief Description

Sample player for positional 2D Sound.

12.273.2 Member Functions


<table>
<thead>
<tr>
<th>type</th>
<th>function</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>get_polyphony ( ) const</td>
</tr>
<tr>
<td>float</td>
<td>get_random_pitch_scale ( ) const</td>
</tr>
<tr>
<td>SampleLibrary</td>
<td>get_sample_library ( ) const</td>
</tr>
<tr>
<td>bool</td>
<td>is_voice_active ( int voice ) const</td>
</tr>
<tr>
<td>int</td>
<td>play ( String sample, int voice=-2 )</td>
</tr>
<tr>
<td>void</td>
<td>set_polyphony ( int max_voices )</td>
</tr>
<tr>
<td>void</td>
<td>set_random_pitch_scale ( float val )</td>
</tr>
<tr>
<td>void</td>
<td>set_sample_library ( SampleLibrary library )</td>
</tr>
<tr>
<td>void</td>
<td>stop_all ( )</td>
</tr>
<tr>
<td>void</td>
<td>stop_voice ( int voice )</td>
</tr>
<tr>
<td>void</td>
<td>voice_set_pitch_scale ( int voice, float ratio )</td>
</tr>
<tr>
<td>void</td>
<td>voice_set_volume_scale_db ( int voice, float db )</td>
</tr>
</tbody>
</table>

12.273.3 Numeric Constants

- INVALID_VOICE = -1 — Value returned if the voice or sample are invalid.

- NEXT_VOICE = -2 — Default voice for the play method. Corresponds to the first voice following the last used voice.
12.273.4 Description

Sample player for positional 2D Sound. Plays sound samples positionally, left and right depending on the distance/place on the screen.

12.273.5 Member Function Description

- `int get_polyphony() const`
  Return the polyphony of the player.

- `float get_random_pitch_scale() const`
  Return the amplitude used for random pitch scale variations.

- `SampleLibrary get_sample_library() const`
  Return the sample library used by the player.

- `bool is_voice_active(int voice) const`
  Return whether a voice is still active or has stopped playing.

- `int play(String sample, int voice=-2)`
  Play a sample. An internal polyphony ID can optionally be passed, or defaults to NEXT_VOICE. Return a voice ID which can be used to modify the voice parameters, or INVALID_VOICE if the voice or sample are invalid.

- `void set_polyphony(int max_voices)`
  Set the polyphony of the player (maximum amount of simultaneous voices).

- `void set_random_pitch_scale(float val)`
  Set the amplitude for random pitch scale variations. If different from zero, the pitch scale will vary randomly around 1.0 in a range defined by val.

  The actual pitch scale will be, with “variation” ranging from -val to val:

  * variation > 0: $1.0 + \text{variation}$
  * variation < 0: $1.0/(1.0 - \text{variation})$

- `void set_sample_library(SampleLibrary library)`
  Set the sample library for the player.

- `void stop_all()`
  Stop all playing voices.

- `void stop_voice(int voice)`
  Stop a given voice.

- `void voice_set_pitch_scale(int voice, float ratio)`
  Change the pitch scale of a currently playing voice.

- `void voice_set_volume_scale_db(int voice, float db)`
  Change the volume scale (in dB) of a currently playing voice.
12.274 SceneState

Inherits: Reference < Object

Category: Core

12.274.1 Brief Description

12.274.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td>get_connection_binds ( int idx ) const</td>
</tr>
<tr>
<td>int</td>
<td>get_connection_count ( ) const</td>
</tr>
<tr>
<td>int</td>
<td>get_connection_flags ( int idx ) const</td>
</tr>
<tr>
<td>String</td>
<td>get_connection_method ( int idx ) const</td>
</tr>
<tr>
<td>String</td>
<td>get_connection_signal ( int idx ) const</td>
</tr>
<tr>
<td>NodePath</td>
<td>get_connection_source ( int idx ) const</td>
</tr>
<tr>
<td>NodePath</td>
<td>get_connection_target ( int idx ) const</td>
</tr>
<tr>
<td>int</td>
<td>get_node_count ( ) const</td>
</tr>
<tr>
<td>StringArray</td>
<td>get_node_groups ( int idx ) const</td>
</tr>
<tr>
<td>PackedScene</td>
<td>get_node_instance ( int idx ) const</td>
</tr>
<tr>
<td>String</td>
<td>get_node_instance_placeholder ( int idx ) const</td>
</tr>
<tr>
<td>String</td>
<td>get_node_name ( int idx ) const</td>
</tr>
<tr>
<td>NodePath</td>
<td>get_node_owner_path ( int idx ) const</td>
</tr>
<tr>
<td>NodePath</td>
<td>get_node_path ( int idx, bool for_parent=false ) const</td>
</tr>
<tr>
<td>int</td>
<td>get_node_property_count ( int idx ) const</td>
</tr>
<tr>
<td>String</td>
<td>get_node_property_name ( int idx, int prop_idx ) const</td>
</tr>
<tr>
<td>void</td>
<td>get_node_property_value ( int idx, int prop_idx ) const</td>
</tr>
<tr>
<td>String</td>
<td>get_node_type ( int idx ) const</td>
</tr>
<tr>
<td>bool</td>
<td>is_node_instance_placeholder ( int idx ) const</td>
</tr>
</tbody>
</table>

12.274.3 Member Function Description

- Array get_connection_binds ( int idx ) const
- int get_connection_count ( ) const
- int get_connection_flags ( int idx ) const
- String get_connection_method ( int idx ) const
- String get_connection_signal ( int idx ) const
- NodePath get_connection_source ( int idx ) const
- NodePath get_connection_target ( int idx ) const
- int get_node_count ( ) const
- StringArray get_node_groups ( int idx ) const
- PackedScene get_node_instance ( int idx ) const
- String get_node_instance_placeholder ( int idx ) const
- String get_node_name ( int idx ) const
- NodePath get_node_owner_path ( int idx ) const
- `NodePath get_node_path ( int idx, bool for_parent=false )` const
- `int get_node_property_count ( int idx )` const
- `String get_node_property_name ( int idx, int prop_idx )` const
- `void get_node_property_value ( int idx, int prop_idx )` const
- `String get_node_type ( int idx )` const
- `bool is_node_instance_placeholder ( int idx )` const

### 12.275 SceneTree

**Inherits:** `MainLoop < Object`

**Category:** Core
12.275.1 Brief Description

12.275.2 Member Functions

<table>
<thead>
<tr>
<th>void</th>
<th>call_group ( int flags, String group, String method, Variant arg0=NULL, Variant arg1=NULL, Variant arg2=NULL, Variant arg3=NULL, Variant arg4=NULL )</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>change_scene ( String path )</td>
</tr>
<tr>
<td>int</td>
<td>change_scene_to ( PackedScene packed_scene )</td>
</tr>
<tr>
<td>Node</td>
<td>get_current_scene ( ) const</td>
</tr>
<tr>
<td>Object</td>
<td>get_editor_root ( ) const</td>
</tr>
<tr>
<td>int</td>
<td>get_frame ( ) const</td>
</tr>
<tr>
<td>int</td>
<td>get_node_count ( ) const</td>
</tr>
<tr>
<td>Array</td>
<td>get_nodes_in_group ( String group )</td>
</tr>
<tr>
<td>Viewport</td>
<td>get_root ( ) const</td>
</tr>
<tr>
<td>bool</td>
<td>has_group ( String name ) const</td>
</tr>
<tr>
<td>bool</td>
<td>is_debugging_collisions_hint ( ) const</td>
</tr>
<tr>
<td>bool</td>
<td>is_debugging_navigation_hint ( ) const</td>
</tr>
<tr>
<td>bool</td>
<td>is_editor_hint ( ) const</td>
</tr>
<tr>
<td>bool</td>
<td>is_input_handled ( )</td>
</tr>
<tr>
<td>bool</td>
<td>is_paused ( ) const</td>
</tr>
<tr>
<td>void</td>
<td>notify_group ( int call_flags, String group, int notification )</td>
</tr>
<tr>
<td>void</td>
<td>queue_delete ( Object obj )</td>
</tr>
<tr>
<td>void</td>
<td>quit ( )</td>
</tr>
<tr>
<td>int</td>
<td>reload_current_scene ( )</td>
</tr>
<tr>
<td>void</td>
<td>set_auto_accept_quit ( bool enabled )</td>
</tr>
<tr>
<td>void</td>
<td>set_current_scene ( Node child_node )</td>
</tr>
<tr>
<td>void</td>
<td>set_debug_collisions_hint ( bool enable )</td>
</tr>
<tr>
<td>void</td>
<td>set_debug_navigation_hint ( bool enable )</td>
</tr>
<tr>
<td>void</td>
<td>set_editor_hint ( bool enable )</td>
</tr>
<tr>
<td>void</td>
<td>set_group ( int call_flags, String group, String property, Variant value )</td>
</tr>
<tr>
<td>void</td>
<td>set_input_as_handled ( )</td>
</tr>
<tr>
<td>void</td>
<td>set_pause ( bool enable )</td>
</tr>
<tr>
<td>void</td>
<td>set_screen_stretch ( int mode, int aspect, Vector2 minsize )</td>
</tr>
</tbody>
</table>

12.275.3 Signals

- files_dropped ( StringArray files, int screen )
- fixed_frame ( )
- idle_frame ( )
- node_configuration_warning_changed ( Object node )
- node_removed ( Object node )
- screen_resized ( )
- tree_changed ( )
12.275.4 Numeric Constants

- GROUP_CALL_DEFAULT = 0
- GROUP_CALL_REVERSE = 1
- GROUP_CALL_REALTIME = 2
- GROUP_CALL_UNIQUE = 4
- STRETCH_MODE_DISABLED = 0
- STRETCH_MODE_2D = 1
- STRETCH_MODE_VIEWPORT = 2
- STRETCH_ASPECT_IGNORE = 0
- STRETCH_ASPECT_KEEP = 1
- STRETCH_ASPECT_KEEP_WIDTH = 2
- STRETCH_ASPECT_KEEP_HEIGHT = 3

12.275.5 Member Function Description

- void call_group ( int flags, String group, String method, Variant arg0==NULL, Variant arg1=NULL, Variant arg2=NULL, Variant arg3=NULL, Variant arg4=NULL )
- int change_scene ( String path )
- int change_scene_to ( PackedScene packed_scene )
- Node get_current_scene ( ) const
- Object get_edited_scene_root ( ) const
- int get_frame ( ) const
- int get_node_count ( ) const
- Array get_nodes_in_group ( String group )
- Viewport get_root ( ) const
- bool has_group ( String name ) const
- bool is_debugging_collisions_hint ( ) const
- bool is_debugging_navigation_hint ( ) const
- bool is_editor_hint ( ) const
- bool is_input_handled ( )
- bool is_paused ( ) const
- void notify_group ( int call_flags, String group, int notification )
- void queue_delete ( Object obj )
- void quit ( )
- int reload_current_scene ( )
- void set_auto_accept_quit ( bool enabled )
- void set_current_scene ( Node child_node )
• void set_debug_collisions_hint ( bool enable )
• void set_debug_navigation_hint ( bool enable )
• void set_edited_scene_root ( Object scene )
• void set_editor_hint ( bool enable )
• void set_group ( int call_flags, String group, String property, Variant value )
• void set_input_as_handled ()
• void set_pause ( bool enable )
• void set_screen_stretch ( int mode, int aspect, Vector2 minsize )

12.276 Script

Inherits: Resource < Reference < Object
Inherited By: GDScript
Category: Core

12.276.1 Brief Description

Base class for scripts.

12.276.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool</td>
<td>can_instance ( ) const</td>
</tr>
<tr>
<td>String</td>
<td>get_source_code ( ) const</td>
</tr>
<tr>
<td>bool</td>
<td>has_source_code ( ) const</td>
</tr>
<tr>
<td>bool</td>
<td>instance_has ( Object base_object ) const</td>
</tr>
<tr>
<td>int</td>
<td>reload ( bool keep_state=false )</td>
</tr>
<tr>
<td>void</td>
<td>set_source_code ( String source )</td>
</tr>
</tbody>
</table>

12.276.3 Description

Base class for scripts. Any script that is loaded becomes one of these resources, which can then create instances.

12.276.4 Member Function Description

• bool can_instance ( ) const
  Return true if this script can be instance (ie not a library).

• String get_source_code ( ) const
  Return the script source code (if available).

• bool has_source_code ( ) const
  Return true if the script contains source code.

• bool instance_has ( Object base_object ) const
Return true if a given object uses an instance of this script.

- `int reload ( bool keep_state=false )`
- `void set_source_code ( String source )`

Set the script source code.

### 12.277 ScrollBar

**Inherits:** Range < Control < CanvasItem < Node < Object

**Inherited By:** HScrollBar, VScrollBar

**Category:** Core

#### 12.277.1 Brief Description

Base class for scroll bars.

#### 12.277.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>float</td>
<td><code>get_custom_step ( ) const</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_custom_step ( float step )</code></td>
</tr>
</tbody>
</table>

#### 12.277.3 Description

Scrollbars are a `Range` based `Control`, that display a draggable area (the size of the page). Horizontal (`HScrollBar`) and Vertical (`VScrollBar`) versions are available.

#### 12.277.4 Member Function Description

- `float get_custom_step ( ) const`
- `void set_custom_step ( float step )`

### 12.278 ScrollContainer

**Inherits:** Container < Control < CanvasItem < Node < Object

**Category:** Core

#### 12.278.1 Brief Description

A helper node for displaying scrollable elements (e.g. lists).
12.278.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>get_h_scroll() const</td>
</tr>
<tr>
<td>int</td>
<td>get_v_scroll() const</td>
</tr>
<tr>
<td>bool</td>
<td>is_h_scroll_enabled() const</td>
</tr>
<tr>
<td>bool</td>
<td>is_v_scroll_enabled() const</td>
</tr>
<tr>
<td>void</td>
<td>set_enable_h_scroll(bool enable)</td>
</tr>
<tr>
<td>void</td>
<td>set_enable_v_scroll(bool enable)</td>
</tr>
<tr>
<td>void</td>
<td>set_h_scroll(int val)</td>
</tr>
<tr>
<td>void</td>
<td>set_v_scroll(int val)</td>
</tr>
</tbody>
</table>

12.278.3 Description

A ScrollContainer node with a Control child and scrollbar child (HScrollbar, VScrollBar, or both) will only draw the Control within the ScrollContainer area. Scrollbars will automatically be drawn at the right (for vertical) or bottom (for horizontal) and will enable dragging to move the viewable Control (and its children) within the ScrollContainer. Scrollbars will also automatically resize the grabber based on the minimum_size of the Control relative to the ScrollContainer. Works great with a Panel control. You can set EXPAND on children size flags, so they will upscale to ScrollContainer size if ScrollContainer size is bigger (scroll is invisible for chosen dimension).

12.278.4 Member Function Description

- int get_h_scroll() const
  Return current horizontal scroll value.

- int get_v_scroll() const
  Return current vertical scroll value.

- bool is_h_scroll_enabled() const
  Return true if horizontal scroll is allowed.

- bool is_v_scroll_enabled() const
  Return true if vertical scroll is allowed.

- void set_enable_h_scroll(bool enable)
  Set allows horizontal scroll.

- void set_enable_v_scroll(bool enable)
  Set allows vertical scroll.

- void set_h_scroll(int val)
  Set horizontal scroll value.

- void set_v_scroll(int val)
  Set vertical scroll value.

12.279 SegmentShape2D

Inherits: Shape2D < Resource < Reference < Object
Category: Core

12.279.1 Brief Description

Segment Shape for 2D Collision Detection.

12.279.2 Member Functions

| Vector2 | get_a ( ) const |
| Vector2 | get_b ( ) const |
| void    | set_a ( Vector2 a ) |
| void    | set_b ( Vector2 b ) |

12.279.3 Description

Segment Shape for 2D Collision Detection, consists of two points, ‘a’ and ‘b’.

12.279.4 Member Function Description

- **Vector2 get_a ( ) const**
  Return the first point’s position.
- **Vector2 get_b ( ) const**
  Return the second point’s position.
- **void set_a ( Vector2 a )**
  Set the first point’s position.
- **void set_b ( Vector2 b )**
  Set the second point’s position.

12.280 Semaphore

Inherits: Reference < Object

Category: Core

12.280.1 Brief Description

A synchronization Semaphore.

12.280.2 Member Functions

| Error  | post ( ) |
| Error  | wait ( ) |
12.280.3 Description

A synchronization Semaphore. Element used in multi-threading. Initialized to zero on creation.

12.280.4 Member Function Description

• Error post()
  Lowers the Semaphore, allowing one more thread in.

• Error wait()
  Tries to wait for the Semaphore, if its value is zero, blocks until non-zero.

12.281 Separator

Inherits: Control < CanvasItem < Node < Object
Inherited By: VSeparator, HSeparator
Category: Core

12.281.1 Brief Description

Base class for separators.

12.281.2 Description

Separator is a Control used for separating other controls. It’s purely a visual decoration. Horizontal (HSeparator) and Vertical (VSeparator) versions are available.

12.282 Shader

Inherits: Resource < Reference < Object
Inherited By: MaterialShader, CanvasItemShader, ShaderGraph
Category: Core

12.282.1 Brief Description

To be changed, ignore.
12.282.2 Member Functions

<table>
<thead>
<tr>
<th>Texture</th>
<th>get_default_texture_param ( String param ) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>get_fragment_code ( ) const</td>
</tr>
<tr>
<td>String</td>
<td>get_light_code ( ) const</td>
</tr>
<tr>
<td>int</td>
<td>get_mode ( ) const</td>
</tr>
<tr>
<td>String</td>
<td>get_vertex_code ( ) const</td>
</tr>
<tr>
<td>bool</td>
<td>has_param ( String name ) const</td>
</tr>
<tr>
<td>void</td>
<td>set_code ( String vcode, String fcode, String lcode, int fofs=0, int lofs=0 )</td>
</tr>
<tr>
<td>void</td>
<td>set_default_texture_param ( String param, Texture texture )</td>
</tr>
</tbody>
</table>

12.282.3 Numeric Constants

- MODE_MATERIAL = 0
- MODE_CANVAS_ITEM = 1
- MODE_POST_PROCESS = 2

12.282.4 Description

To be changed, ignore.

12.282.5 Member Function Description

- Texture get_default_texture_param ( String param ) const
- String get_fragment_code ( ) const
- String get_light_code ( ) const
- int get_mode ( ) const
- String get_vertex_code ( ) const
- bool has_param ( String name ) const
- void set_code ( String vcode, String fcode, String lcode, int fofs=0, int lofs=0 )
- void set_default_texture_param ( String param, Texture texture )

12.283 ShaderGraph

Inherits: Shader < Resource < Reference < Object

Inherited By: MaterialShaderGraph, CanvasItemShaderGraph

Category: Core

12.283.1 Brief Description

12.283.2 Member Functions
### Method Definitions

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>clear(int shader_type)</code></td>
<td></td>
</tr>
<tr>
<td><code>ColorArray color_ramp_node_get_colors(int shader_type, int id)</code></td>
<td>Const</td>
</tr>
<tr>
<td><code>RealArray color_ramp_node_get_offsets(int shader_type, int id)</code></td>
<td>Const</td>
</tr>
<tr>
<td><code>void color_ramp_node_set_ramp(int shader_type, int id, ColorArray colors, RealArray offsets)</code></td>
<td></td>
</tr>
<tr>
<td><code>String comment_node_get_text(int shader_type, int id)</code></td>
<td>Const</td>
</tr>
<tr>
<td><code>void comment_node_set_text(int shader_type, int id, String text)</code></td>
<td></td>
</tr>
<tr>
<td><code>Error connect_node(int shader_type, int src_id, int src_slot, int dst_id, int dst_slot)</code></td>
<td></td>
</tr>
<tr>
<td><code>CubeMap cubemap_input_node_get_value(int shader_type, int id)</code></td>
<td>Const</td>
</tr>
<tr>
<td><code>void cubemap_input_node_set_value(int shader_type, int id, CubeMap value)</code></td>
<td></td>
</tr>
<tr>
<td><code>Vector2Array curve_map_node_get_points(int shader_type, int id)</code></td>
<td>Const</td>
</tr>
<tr>
<td><code>void curve_map_node_set_points(int shader_type, int id, Vector2Array points)</code></td>
<td></td>
</tr>
<tr>
<td><code>void default_get_value(int shader_type, int id, int param_id)</code></td>
<td></td>
</tr>
<tr>
<td><code>void default_set_value(int shader_type, int id, int param_id, Variant value)</code></td>
<td></td>
</tr>
<tr>
<td><code>void disconnect_node(int shader_type, int id, int src_slot, int dst_id, int dst_slot)</code></td>
<td></td>
</tr>
<tr>
<td><code>Array get_node_connections(int shader_type)</code></td>
<td>Const</td>
</tr>
<tr>
<td><code>Array get_node_list(int shader_type)</code></td>
<td>Const</td>
</tr>
<tr>
<td><code>String input_node_get_name(int shader_type, int id)</code></td>
<td></td>
</tr>
<tr>
<td><code>void input_node_set_name(int shader_type, int id, String name)</code></td>
<td></td>
</tr>
<tr>
<td><code>bool is_node_connected(int shader_type, int src_id, int src_slot, int dst_id, int dst_slot)</code></td>
<td></td>
</tr>
<tr>
<td><code>void node_add(int shader_type, int node_type, int id)</code></td>
<td>Const</td>
</tr>
<tr>
<td><code>Vector2 node_get_pos(int shader_type, int id)</code></td>
<td>Const</td>
</tr>
<tr>
<td><code>Variant node_get_state(int shader_type, int id)</code></td>
<td>Const</td>
</tr>
<tr>
<td><code>int node_get_type(int shader_type, int id)</code></td>
<td>Const</td>
</tr>
<tr>
<td><code>void node_remove(int shader_type, int id)</code></td>
<td></td>
</tr>
<tr>
<td><code>void node_set_pos(int shader_type, int id, Vector2 pos)</code></td>
<td></td>
</tr>
<tr>
<td><code>void node_set_state(int shader_type, int id, Variant state)</code></td>
<td></td>
</tr>
<tr>
<td><code>Color rgb_const_node_get_value(int shader_type, int id)</code></td>
<td>Const</td>
</tr>
<tr>
<td><code>void rgb_const_node_set_value(int shader_type, int id, Color value)</code></td>
<td></td>
</tr>
<tr>
<td><code>Color rgb_input_node_get_value(int shader_type, int id)</code></td>
<td></td>
</tr>
<tr>
<td><code>void rgb_input_node_set_value(int shader_type, int id, Color value)</code></td>
<td></td>
</tr>
<tr>
<td><code>int rgb_op_node_get_op(int shader_type, float id)</code></td>
<td>Const</td>
</tr>
<tr>
<td><code>void rgb_op_node_set_op(int shader_type, float id, int op)</code></td>
<td></td>
</tr>
<tr>
<td><code>float scalar_const_node_get_value(int shader_type, int id)</code></td>
<td>Const</td>
</tr>
<tr>
<td><code>void scalar_const_node_set_value(int shader_type, int id, float value)</code></td>
<td></td>
</tr>
<tr>
<td><code>int scalar_func_node_get_function(int shader_type, int id)</code></td>
<td>Const</td>
</tr>
<tr>
<td><code>void scalar_func_node_set_function(int shader_type, int id, int func)</code></td>
<td></td>
</tr>
<tr>
<td><code>float scalar_input_node_get_value(int shader_type, int id)</code></td>
<td>Const</td>
</tr>
<tr>
<td><code>void scalar_input_node_set_value(int shader_type, int id, float value)</code></td>
<td></td>
</tr>
<tr>
<td><code>int scalar_op_node_get_op(int shader_type, float id)</code></td>
<td>Const</td>
</tr>
<tr>
<td><code>void scalar_op_node_set_op(int shader_type, float id, int op)</code></td>
<td></td>
</tr>
<tr>
<td><code>Texture texture_input_node_get_value(int shader_type, int id)</code></td>
<td>Const</td>
</tr>
<tr>
<td><code>void texture_input_node_set_value(int shader_type, int id, Texture value)</code></td>
<td></td>
</tr>
<tr>
<td><code>int texture_node_get_filter_size(int shader_type, int id)</code></td>
<td>Const</td>
</tr>
<tr>
<td><code>float texture_node_get_filter_strength(int shader_type, float id)</code></td>
<td>Const</td>
</tr>
<tr>
<td><code>void texture_node_set_filter_size(int shader_type, int id, int filter_size)</code></td>
<td></td>
</tr>
<tr>
<td><code>void texture_node_set_filter_strength(int shader_type, float id, float filter_strength)</code></td>
<td></td>
</tr>
<tr>
<td><code>Vector3 vec_const_node_get_value(int shader_type, int id)</code></td>
<td>Const</td>
</tr>
<tr>
<td><code>void vec_const_node_set_value(int shader_type, int id, Vector3 value)</code></td>
<td></td>
</tr>
<tr>
<td><code>int vec_func_node_get_function(int shader_type, int id)</code></td>
<td>Const</td>
</tr>
<tr>
<td><code>void vec_func_node_set_function(int shader_type, int id, int func)</code></td>
<td></td>
</tr>
</tbody>
</table>

---

Continued on next page
Table 12.33 – continued from previous page

<table>
<thead>
<tr>
<th>Class</th>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vector3</td>
<td>vec_input_node_get_value</td>
<td>(int shader_type, int id) const</td>
</tr>
<tr>
<td></td>
<td>vec_input_node_set_value</td>
<td>(int shader_type, int id, Vector3 value)</td>
</tr>
<tr>
<td>int</td>
<td>vec_op_node_get_op</td>
<td>(int shader_type, float id) const</td>
</tr>
<tr>
<td>void</td>
<td>vec_op_node_set_op</td>
<td>(int shader_type, float id, int op)</td>
</tr>
<tr>
<td>int</td>
<td>vec_scalar_op_node_get_op</td>
<td>(int shader_type, float id) const</td>
</tr>
<tr>
<td>void</td>
<td>vec_scalar_op_node_set_op</td>
<td>(int shader_type, float id, int op)</td>
</tr>
<tr>
<td>Transform</td>
<td>xform_const_node_get_value</td>
<td>(int shader_type, int id) const</td>
</tr>
<tr>
<td>void</td>
<td>xform_const_node_set_value</td>
<td>(int shader_type, int id, Transform value)</td>
</tr>
<tr>
<td>Transform</td>
<td>xform_input_node_get_value</td>
<td>(int shader_type, int id) const</td>
</tr>
<tr>
<td>void</td>
<td>xform_input_node_set_value</td>
<td>(int shader_type, int id, Transform value)</td>
</tr>
<tr>
<td>bool</td>
<td>xform_vec_mult_node_get_no_translation</td>
<td>(int shader_type, int id) const</td>
</tr>
<tr>
<td>void</td>
<td>xform_vec_mult_node_set_no_translation</td>
<td>(int shader_type, int id, bool disable)</td>
</tr>
</tbody>
</table>

12.283.3 Signals

- updated ( )

12.283.4 Numeric Constants

- NODE_INPUT = 0
- NODE_SCALAR_CONST = 1
- NODE_VEC_CONST = 2
- NODE_RGB_CONST = 3
- NODE_XFORM_CONST = 4
- NODE_TIME = 5
- NODE_SCREEN_TEX = 6
- NODE_SCALAR_OP = 7
- NODE_VEC_OP = 8
- NODE_VEC_SCALAR_OP = 9
- NODE_RGB_OP = 10
- NODE_XFORM_MULT = 11
- NODE_XFORM_VEC_MULT = 12
- NODE_XFORM_VEC_INV_MULT = 13
- NODE_SCALAR_FUNC = 14
- NODE_VEC_FUNC = 15
- NODE_VEC_LEN = 16
- NODE_DOT_PROD = 17
- NODE_VEC_TO_SCALAR = 18
- NODE_SCALAR_TO_VEC = 19
- NODE_VEC_TO_XFORM = 21
• NODE_XFORM_TO_VEC = 20
• NODE_SCALAR_INTERP = 22
• NODE_VEC_INTERP = 23
• NODE_COLOR_RAMP = 24
• NODE_CURVE_MAP = 25
• NODE_SCALAR_INPUT = 26
• NODE_VEC_INPUT = 27
• NODE_RGB_INPUT = 28
• NODE_XFORM_INPUT = 29
• NODE_TEXTURE_INPUT = 30
• NODE_CUBEMAP_INPUT = 31
• NODE_DEFAULT_TEXTURE = 32
• NODE_OUTPUT = 33
• NODE_COMMENT = 34
• NODE_TYPE_MAX = 35
• SLOT_TYPE_SCALAR = 0
• SLOT_TYPE_VEC = 1
• SLOT_TYPE_XFORM = 2
• SLOT_TYPE_TEXTURE = 3
• SLOT_MAX = 4
• SHADER_TYPE_VERTEX = 0
• SHADER_TYPE_FRAGMENT = 1
• SHADER_TYPE_LIGHT = 2
• SHADER_TYPE_MAX = 3
• SLOT_IN = 0
• SLOT_OUT = 1
• GRAPH_OK = 0
• GRAPH_ERROR_CYCLIC = 1
• GRAPH_ERROR_MISSING_CONNECTIONS = 2
• SCALAR_OP_ADD = 0
• SCALAR_OP_SUB = 1
• SCALAR_OP_MUL = 2
• SCALAR_OP_DIV = 3
• SCALAR_OP_MOD = 4
• SCALAR_OP_POW = 5
• SCALAR_OP_MAX = 6
• SCALAR_OP_MIN = 7
• SCALAR_OP_ATAN2 = 8
• SCALAR_MAX_OP = 9
• VEC_OP_ADD = 0
• VEC_OP_SUB = 1
• VEC_OP_MUL = 2
• VEC_OP_DIV = 3
• VEC_OP_MOD = 4
• VEC_OP_POW = 5
• VEC_OP_MAX = 6
• VEC_OP_MIN = 7
• VEC_OP_CROSS = 8
• VEC_MAX_OP = 9
• VEC_SCALAR_OP_MUL = 0
• VEC_SCALAR_OP_DIV = 1
• VEC_SCALAR_OP_POW = 2
• VEC_SCALAR_MAX_OP = 3
• RGB_OP_SCREEN = 0
• RGB_OP_DIFFERENCE = 1
• RGB_OP_DARKEN = 2
• RGB_OP_LIGHTEN = 3
• RGB_OP_OVERLAY = 4
• RGB_OP_DODGE = 5
• RGB_OP_BURN = 6
• RGB_OP_SOFT_LIGHT = 7
• RGB_OP_HARD_LIGHT = 8
• RGB_MAX_OP = 9
• SCALAR_FUNC_SIN = 0
• SCALAR_FUNC_COS = 1
• SCALAR_FUNC_TAN = 2
• SCALAR_FUNC_ASIN = 3
• SCALAR_FUNC_ACOS = 4
• SCALAR_FUNC_ATAN = 5
• SCALAR_FUNC_SINH = 6
• SCALAR_FUNC_COSH = 7
• SCALAR_FUNC_TANH = 8
• SCALAR_FUNC_LOG = 9
• SCALAR_FUNC_EXP = 10
• SCALAR_FUNC_SQRT = 11
• SCALAR_FUNC_ABS = 12
• SCALAR_FUNC_SIGN = 13
• SCALAR_FUNC_FLOOR = 14
• SCALAR_FUNC_ROUND = 15
• SCALAR_FUNC_CEIL = 16
• SCALAR_FUNC_FRAC = 17
• SCALAR_FUNC_SATURATE = 18
• SCALAR_FUNC_NEGATE = 19
• SCALAR_MAX_FUNC = 20
• VEC_FUNC_NORMALIZE = 0
• VEC_FUNC_SATURATE = 1
• VEC_FUNC_NEGATE = 2
• VEC_FUNC_RECIPROCAL = 3
• VEC_FUNC_RGB2HSV = 4
• VEC_FUNC_HSV2RGB = 5
• VEC_MAX_FUNC = 6

12.283.5 Member Function Description

• void clear ( int shader_type )
• ColorArray color_ramp_node_get_colors ( int shader_type, int id ) const
• RealArray color_ramp_node_get_offsets ( int shader_type, int id ) const
• void color_ramp_node_set_ramp ( int shader_type, int id, ColorArray colors, RealArray offsets )
• String comment_node_get_text ( int shader_type, int id ) const
• void comment_node_set_text ( int shader_type, int id, String text )
• Error connect_node ( int shader_type, int src_id, int src_slot, int dst_id, int dst_slot )
• CubeMap cubemap_input_node_get_value ( int shader_type, int id ) const
• void cubemap_input_node_set_value ( int shader_type, int id, CubeMap value )
• Vector2Array curve_map_node_get_points ( int shader_type, int id ) const
• void curve_map_node_set_points ( int shader_type, int id, Vector2Array points )
• void default_get_value ( int shader_type, int id, int param_id )
• void default_set_value ( int shader_type, int id, int param_id, Variant value )
• void disconnect_node ( int shader_type, int src_id, int src_slot, int dst_id, int dst_slot )
• Array get_node_connections ( int shader_type ) const
• `Array get_node_list (int shader_type) const`
• `String input_node_get_name (int shader_type, int id)`
• `void input_node_set_name (int shader_type, int id, String name)`
• `bool is_node_connected (int shader_type, int src_id, int src_slot, int dst_id, int dst_slot) const`
• `void node_add (int shader_type, int node_type, int id)`
• `Vector2 node_get_pos (int shader_type, int id) const`
• `Variant node_get_state (int shader_type, int id) const`
• `int node_get_type (int shader_type, int id) const`
• `void node_remove (int shader_type, int id)`
• `void node_set_pos (int shader_type, int id, Vector2 pos)`
• `void node_set_state (int shader_type, int id, Variant state)`
• `Color rgb_const_node_get_value (int shader_type, int id) const`
• `void rgb_const_node_set_value (int shader_type, int id, Color value)`
• `Color rgb_input_node_get_value (int shader_type, int id) const`
• `void rgb_input_node_set_value (int shader_type, int id, Color value)`
• `int rgb_op_node_get_op (int shader_type, float id) const`
• `void rgb_op_node_set_op (int shader_type, float id, int op)`
• `float scalar_const_node_get_value (int shader_type, int id) const`
• `void scalar_const_node_set_value (int shader_type, int id, float value)`
• `int scalar_func_node_get_function (int shader_type, int id) const`
• `void scalar_func_node_set_function (int shader_type, int id, int func)`
• `float scalar_input_node_get_value (int shader_type, int id) const`
• `void scalar_input_node_set_value (int shader_type, int id, float value)`
• `int scalar_op_node_get_op (int shader_type, float id) const`
• `void scalar_op_node_set_op (int shader_type, float id, int op)`
• `Texture texture_input_node_get_value (int shader_type, int id) const`
• `void texture_input_node_set_value (int shader_type, int id, Texture value)`
• `int texture_node_get_filter_size (int shader_type, int id) const`
• `float texture_node_get_filter_strength (int shader_type, float id) const`
• `void texture_node_set_filter_size (int shader_type, int id, int filter_size)`
• `void texture_node_set_filter_strength (int shader_type, float id, float filter_strength)`
• `Vector3 vec_const_node_get_value (int shader_type, int id) const`
• `void vec_const_node_set_value (int shader_type, int id, Vector3 value)`
• `int vec_func_node_get_function (int shader_type, int id) const`
• `void vec_func_node_set_function (int shader_type, int id, int func)`
• `Vector3 vec_input_node_get_value (int shader_type, int id) const`
12.284 ShaderMaterial

Inherits: Material < Resource < Reference < Object

Category: Core

12.284.1 Brief Description

12.284.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shader</td>
<td>get_shader() const</td>
</tr>
<tr>
<td>Variant</td>
<td>get_shader_param(String param) const</td>
</tr>
<tr>
<td>void</td>
<td>set_shader(Shader shader)</td>
</tr>
<tr>
<td>void</td>
<td>set_shader_param(String param, Variant value)</td>
</tr>
</tbody>
</table>

12.284.3 Member Function Description

• Shader get_shader() const

• Variant get_shader_param(String param) const

• void set_shader(Shader shader)

• void set_shader_param(String param, Variant value)

12.285 Shape

Inherits: Resource < Reference < Object

Inherited By: SphereShape, PlaneShape, CapsuleShape, BoxShape, ConvexPolygonShape, RayShape, ConcavePolygonShape

Category: Core
12.285.1 Brief Description

12.286 Shape2D

Inherits: Resource < Reference < Object

Inherited By: RayShape2D, CapsuleShape2D, LineShape2D, CircleShape2D, ConvexPolygonShape2D, ConcavePolygonShape2D, RectangleShape2D, SegmentShape2D

Category: Core

12.286.1 Brief Description

Base class for all 2D Shapes.

12.286.2 Member Functions

<table>
<thead>
<tr>
<th>bool</th>
<th>collide ( Matrix32 local_xform, Shape2D with_shape, Matrix32 shape_xform )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variant</td>
<td>collide_and_get_contacts ( Matrix32 local_xform, Shape2D with_shape, Matrix32 shape_xform )</td>
</tr>
<tr>
<td>bool</td>
<td>collide_with_motion ( Matrix32 local_xform, Vector2 local_motion, Shape2D with_shape, Matrix32 shape_xform, Vector2 shape_motion )</td>
</tr>
<tr>
<td>Variant</td>
<td>collide_with_motion_and_get_contacts ( Matrix32 local_xform, Vector2 local_motion, Shape2D with_shape, Matrix32 shape_xform, Vector2 shape_motion )</td>
</tr>
<tr>
<td>float</td>
<td>get_custom_solver_bias ( ) const</td>
</tr>
<tr>
<td>void</td>
<td>set_custom_solver_bias ( float bias )</td>
</tr>
</tbody>
</table>

12.286.3 Description

Base class for all 2D Shapes. All 2D shape types inherit from this.

12.286.4 Member Function Description

- **bool collide ( Matrix32 local_xform, Shape2D with_shape, Matrix32 shape_xform )**

Return whether this shape is colliding with another.

This method needs the transformation matrix for this shape (local_xform), the shape to check collisions with (with_shape), and the transformation matrix of that shape (shape_xform).

- **Variant collide_and_get_contacts ( Matrix32 local_xform, Shape2D with_shape, Matrix32 shape_xform )**

Return a list of the points where this shape touches another. If there are no collisions, the list is empty.

This method needs the transformation matrix for this shape (local_xform), the shape to check collisions with (with_shape), and the transformation matrix of that shape (shape_xform).

- **bool collide_with_motion ( Matrix32 local_xform, Vector2 local_motion, Shape2D with_shape, Matrix32 shape_xform, Vector2 shape_motion )**

Return whether this shape would collide with another, if a given movement was applied.
This method needs the transformation matrix for this shape (local_xform), the movement to test on this shape
(local_motion), the shape to check collisions with (with_shape), the transformation matrix of that shape
(shape_xform), and the movement to test onto the other object (shape_motion).

- **Variant collide_with_motion_and_get_contacts ( Matrix32 local_xform, Vector2 local_motion, Shape2D
  with_shape, Matrix32 shape_xform, Vector2 shape_motion )**

  Return a list of the points where this shape would touch another, if a given movement was applied. If there are no
  collisions, the list is empty.

  This method needs the transformation matrix for this shape (local_xform), the movement to test on this shape
  (local_motion), the shape to check collisions with (with_shape), the transformation matrix of that shape
  (shape_xform), and the movement to test onto the other object (shape_motion).

- **float get_custom_solver_bias ( ) const**

  Return the custom solver bias.

- **void set_custom_solver_bias ( float bias )**

  Use a custom solver bias. No need to change this unless you really know what you are doing.

  The solver bias is a factor controlling how much two objects “rebound” off each other, when colliding, to avoid them
  getting into each other because of numerical imprecision.

### 12.287 ShortCut

**Inherits:** Resource < Reference < Object

**Category:** Core

#### 12.287.1 Brief Description

#### 12.287.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>get_as_text ( ) const</td>
</tr>
<tr>
<td>InputEvent</td>
<td>get_shortcut ( ) const</td>
</tr>
<tr>
<td>bool</td>
<td>is_shortcut ( InputEvent event ) const</td>
</tr>
<tr>
<td>bool</td>
<td>is_valid ( ) const</td>
</tr>
<tr>
<td>void</td>
<td>set_shortcut ( InputEvent event )</td>
</tr>
</tbody>
</table>

#### 12.287.3 Member Function Description

- **String get_as_text ( ) const**

- **InputEvent get_shortcut ( ) const**

- **bool is_shortcut ( InputEvent event ) const**

- **bool is_valid ( ) const**

- **void set_shortcut ( InputEvent event )**
12.288 Skeleton

Inherits: Spatial < Node < Object

Category: Core

12.288.1 Brief Description

Skeleton for characters and animated objects.

12.288.2 Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void <code>add_bone</code> (String name)</td>
<td>Add a bone, with name “name”. <code>get_bone_count</code> will become the bone index.</td>
</tr>
<tr>
<td>void <code>bind_child_node_to_bone</code> (int bone_idx, Node node)</td>
<td>Bind child node to bone by bone index and node.</td>
</tr>
<tr>
<td>void <code>clear_bones</code> (</td>
<td>Clear all bones.</td>
</tr>
<tr>
<td>int <code>find_bone</code> (String name) const</td>
<td>Find bone by name, return bone index.</td>
</tr>
<tr>
<td>int <code>get_bone_count</code> (</td>
<td>Get the total number of bones.</td>
</tr>
<tr>
<td>Transform <code>get_bone_custom_pose</code> (int bone_idx) const</td>
<td>Get custom pose of bone by index.</td>
</tr>
<tr>
<td>Transform <code>get_bone_global_pose</code> (int bone_idx) const</td>
<td>Get global pose of bone by index.</td>
</tr>
<tr>
<td>String <code>get_bone_name</code> (int bone_idx) const</td>
<td>Get name of bone by index.</td>
</tr>
<tr>
<td>int <code>get_bone_parent</code> (int bone_idx) const</td>
<td>Get parent bone index of bone by index.</td>
</tr>
<tr>
<td>Transform <code>get_bone_pose</code> (int bone_idx) const</td>
<td>Get pose of bone by index.</td>
</tr>
<tr>
<td>Transform <code>get_bone_rest</code> (int bone_idx) const</td>
<td>Get rest pose of bone by index.</td>
</tr>
<tr>
<td>Array <code>get_bound_child_nodes_to_bone</code> (int bone_idx) const</td>
<td>Get child nodes bound to bone by index.</td>
</tr>
<tr>
<td>bool <code>is_bone_rest_disabled</code> (int bone_idx) const</td>
<td>Check if bone is rest disabled by index.</td>
</tr>
<tr>
<td>void <code>set_bone_custom_pose</code> (int bone_idx, Transform custom_pose)</td>
<td>Set custom pose of bone by index and custom pose.</td>
</tr>
<tr>
<td>void <code>set_bone_disable_rest</code> (int bone_idx, bool disable)</td>
<td>Set rest disable of bone by index and disabled flag.</td>
</tr>
<tr>
<td>void <code>set_bone_global_pose</code> (int bone_idx, Transform pose)</td>
<td>Set global pose of bone by index and pose.</td>
</tr>
<tr>
<td>void <code>set_bone_parent</code> (int bone_idx, int parent_idx)</td>
<td>Set parent of bone by index and parent index.</td>
</tr>
<tr>
<td>void <code>set_bone_pose</code> (int bone_idx, Transform pose)</td>
<td>Set pose of bone by index and pose.</td>
</tr>
<tr>
<td>void <code>set_bone_rest</code> (int bone_idx, Transform rest)</td>
<td>Set rest pose of bone by index and rest pose.</td>
</tr>
<tr>
<td>void <code>unbind_child_node_from_bone</code> (int bone_idx, Node node)</td>
<td>Unbind child node from bone by index and node.</td>
</tr>
<tr>
<td>void <code>unparent_bone_and_rest</code> (int bone_idx)</td>
<td>Unparent bone and set rest to default.</td>
</tr>
</tbody>
</table>

12.288.3 Numeric Constants

- `NOTIFICATION_UPDATE_SKELETON = 50`

12.288.4 Description

Skeleton provides a hierarchical interface for managing bones, including pose, rest and animation (see Animation). Skeleton will support rag doll dynamics in the future.

12.288.5 Member Function Description

- void `add_bone` (String name)

  Add a bone, with name “name”. `get_bone_count` will become the bone index.
• void bind_child_node_to_bone ( int bone_idx, Node node )

Deprecated soon.

• void clear_bones ( )

Clear all the bones in this skeleton.

• int find_bone ( String name ) const

Return the bone index that matches “name” as its name.

• int get_bone_count ( ) const

Return the amount of bones in the skeleton.

• Transform get_bone_custom_pose ( int bone_idx ) const

• Transform get_bone_global_pose ( int bone_idx ) const

• String get_bone_name ( int bone_idx ) const

Return the name of the bone at index “index”

• int get_bone_parent ( int bone_idx ) const

Return the bone index which is the parent of the bone at “bone_idx”. If -1, then bone has no parent. Note that the
parent bone returned will always be less than “bone_idx”.

• Transform get_bone_pose ( int bone_idx ) const

Return the pose transform for bone “bone_idx”.

• Transform get_bone_rest ( int bone_idx ) const

Return the rest transform for a bone “bone_idx”.

• Transform get_bone_transform ( int bone_idx ) const

• Array get_bound_child_nodes_to_bone ( int bone_idx ) const

Deprecated soon.

• bool is_bone_rest_disabled ( int bone_idx ) const

• void set_bone_custom_pose ( int bone_idx, Transform custom_pose )

• void set_bone_disable_rest ( int bone_idx, bool disable )

• void set_bone_global_pose ( int bone_idx, Transform pose )

• void set_bone_parent ( int bone_idx, int parent_idx )

Set the bone index “parent_idx” as the parent of the bone at “bone_idx”. If -1, then bone has no parent. Note: “parent_idx” must be less than “bone_idx”.

• void set_bone_pose ( int bone_idx, Transform pose )

Return the pose transform for bone “bone_idx”.

• void set_bone_rest ( int bone_idx, Transform rest )

Set the rest transform for bone “bone_idx”

• void unbind_child_node_from_bone ( int bone_idx, Node node )

Deprecated soon.

• void unparent_bone_and_rest ( int bone_idx )

• void unparent_bone_and_rest ( int bone_idx )
12.289 Slider

Inherits: Range < Control < CanvasItem < Node < Object
Inherited By: HSlider, VSlider
Category: Core

12.289.1 Brief Description

Base class for GUI Sliders.

12.289.2 Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int get_ticks</td>
<td>Return amounts of ticks to display on slider.</td>
</tr>
<tr>
<td>bool get_ticks_on_borders</td>
<td>Return true if ticks are visible on borders.</td>
</tr>
<tr>
<td>void set_ticks (int count)</td>
<td>Set amount of ticks to display in slider.</td>
</tr>
<tr>
<td>void set_ticks_on_borders (bool ticks_on_border)</td>
<td>Set true if ticks are visible on borders.</td>
</tr>
</tbody>
</table>

12.289.3 Description

Base class for GUI Sliders.

12.289.4 Member Function Description

- **int get_ticks ( ) const**
  Return amounts of ticks to display on slider.
- **bool get_ticks_on_borders ( ) const**
  Return true if ticks are visible on borders.
- **void set_ticks (int count)**
  Set amount of ticks to display in slider.
- **void set_ticks_on_borders (bool ticks_on_border)**
  Set true if ticks are visible on borders.

12.290 SliderJoint

Inherits: Joint < Spatial < Node < Object
Category: Core
12.290.1 Brief Description

12.290.2 Member Functions

<table>
<thead>
<tr>
<th></th>
<th>Function Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>float</td>
<td><code>get_param ( int param ) const</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_param ( int param, float value )</code></td>
</tr>
</tbody>
</table>

12.290.3 Numeric Constants

- PARAM_LINEAR_LIMIT_UPPER = 0
- PARAM_LINEAR_LIMIT_LOWER = 1
- PARAM_LINEAR_LIMIT_SOFTNESS = 2
- PARAM_LINEAR_LIMIT_RESTITUTION = 3
- PARAM_LINEAR_LIMIT_DAMPING = 4
- PARAM_LINEAR_MOTION_SOFTNESS = 5
- PARAM_LINEAR_MOTION_RESTITUTION = 6
- PARAM_LINEAR_MOTION_DAMPING = 7
- PARAM_LINEAR_ORTHOGONAL_SOFTNESS = 8
- PARAM_LINEAR_ORTHOGONAL_RESTITUTION = 9
- PARAM_LINEAR_ORTHOGONAL_DAMPING = 10
- PARAM_ANGULAR_LIMIT_UPPER = 11
- PARAM_ANGULAR_LIMIT_LOWER = 12
- PARAM_ANGULAR_LIMIT_SOFTNESS = 13
- PARAM_ANGULAR_LIMIT_RESTITUTION = 14
- PARAM_ANGULAR_LIMIT_DAMPING = 15
- PARAM_ANGULAR_MOTION_SOFTNESS = 16
- PARAM_ANGULAR_MOTION_RESTITUTION = 17
- PARAM_ANGULAR_MOTION_DAMPING = 18
- PARAM_ANGULAR_ORTHOGONAL_SOFTNESS = 19
- PARAM_ANGULAR_ORTHOGONAL_RESTITUTION = 20
- PARAM_ANGULAR_ORTHOGONAL_DAMPING = 21
- PARAM_MAX = 22

12.290.4 Member Function Description

- float `get_param ( int param ) const`
- void `set_param ( int param, float value )`
12.291 SoundPlayer2D

Inherits: Node2D < CanvasItem < Node < Object

Inherited By: SamplePlayer2D

Category: Core

12.291.1 Brief Description

Base class for playing spatial 2D sound.

12.291.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>float</td>
<td>get_param ( int param ) const</td>
</tr>
<tr>
<td>void</td>
<td>set_param ( int param, float value )</td>
</tr>
</tbody>
</table>

12.291.3 Numeric Constants

- PARAM_VOLUME_DB = 0
- PARAM_PITCH_SCALE = 1
- PARAM_ATTENUATION_MIN_DISTANCE = 2
- PARAM_ATTENUATION_MAX_DISTANCE = 3
- PARAM_ATTENUATION_DISTANCE_EXP = 4
- PARAM_MAX = 5

12.291.4 Description

Base class for playing spatial 2D sound.

12.291.5 Member Function Description

- float get_param ( int param ) const
- void set_param ( int param, float value )

12.292 SoundRoomParams

Inherits: Node < Object

Category: Core
12.292.1 Brief Description

12.292.2 Member Functions

<table>
<thead>
<tr>
<th></th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>float</td>
<td><code>get_param (int param) const</code></td>
</tr>
<tr>
<td>int</td>
<td><code>get_reverb_mode () const</code></td>
</tr>
<tr>
<td>bool</td>
<td><code>is_forcing_params_to_all_sources ()</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_force_params_to_all_sources (bool enabled)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_param (int param, float value)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_reverb_mode (int reverb_mode)</code></td>
</tr>
</tbody>
</table>

12.292.3 Member Function Description

- `float get_param (int param) const`
- `int get_reverb_mode () const`
- `bool is_forcing_params_to_all_sources ()`
- `void set_force_params_to_all_sources (bool enabled)`
- `void set_param (int param, float value)`
- `void set_reverb_mode (int reverb_mode)`

12.293 Spatial

Inherits: `Node < Object`

Inherited By: `Joint, RayCast, Camera, BoneAttachment, CollisionShape, Path, VisualInstance, VehicleWheel, Position3D, ProximityGroup, SpatialPlayer, WorldEnvironment, PathFollow, NavigationMeshInstance, Listener, InverseKinematics, VisibilityNotifier, Navigation, CollisionPolygon, GridMap, Skeleton, CollisionObject`

Category: Core

12.293.1 Brief Description

Base class for all 3D nodes.

12.293.2 Member Functions

<table>
<thead>
<tr>
<th>SpatialGizmo</th>
<th>get_gizmo () const</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transform</td>
<td>get_global_transform () const</td>
</tr>
<tr>
<td>Object</td>
<td>get_parent.spatial () const</td>
</tr>
<tr>
<td>Vector3</td>
<td>get_rotation () const</td>
</tr>
<tr>
<td>Vector3</td>
<td>get_rotation_deg () const</td>
</tr>
<tr>
<td>Vector3</td>
<td>get_scale () const</td>
</tr>
<tr>
<td>Transform</td>
<td>get_transform () const</td>
</tr>
<tr>
<td>Vector3</td>
<td>get_translation () const</td>
</tr>
<tr>
<td>World</td>
<td>get_world () const</td>
</tr>
</tbody>
</table>

Continued on next page
### 12.293.3 Signals

- visibility_changed ( )

### 12.293.4 Numeric Constants

- \textbf{NOTIFICATION\_TRANSFORM\_CHANGED} = 29 — Spatial nodes receive this notification with their global transform changes. This means that either the current or a parent node changed its transform.

- \textbf{NOTIFICATION\_ENTER\_WORLD} = 41

- \textbf{NOTIFICATION\_EXIT\_WORLD} = 42

- \textbf{NOTIFICATION\_VISIBILITY\_CHANGED} = 43

### 12.293.5 Description

Spatial is the base for every type of 3D \textit{Node}. It contains a 3D \textit{Transform} which can be set or get as local or global. If a Spatial \textit{Node} has Spatial children, their transforms will be relative to the parent.
12.293.6 Member Function Description

- **SpatialGizmo** `get_gizmo()` const
- **Transform** `get_global_transform()` const

Return the global transform, relative to worldspace.

- **Object** `get_parent.spatial()` const

Return the parent *Spatial*, or an empty *Object* if no parent exists or parent is not of type *Spatial*.

- **Vector3** `get_rotation()` const
- **Vector3** `get_rotation_deg()` const
- **Vector3** `get_scale()` const
- **Transform** `get_transform()` const

Return the local transform, relative to the bone parent.

- **Vector3** `get_translation()` const
- **World** `get_world()` const
- void `global_rotate` (Vector3 normal, float radians)
- void `global_translate` (Vector3 offset)
- void `hide`
- bool `is_hidden()` const
- bool `is_local_transform_notification_enabled()` const
- bool `is_set_as_toplevel()` const
- bool `is_visible()` const
- void `look_at` (Vector3 target, Vector3 up)
- void `look_at_from_pos` (Vector3 pos, Vector3 target, Vector3 up)
- void `orthonormalize`
- void `rotate` (Vector3 normal, float radians)
- void `rotate_x` (float radians)
- void `rotate_y` (float radians)
- void `rotate_z` (float radians)
- void `set_as_toplevel` (bool enable)
- void `set_gizmo` (SpatialGizmo gizmo)
- void `set_global_transform` (Transform global)

Set the transform globally, relative to worldspace.

- void `set_hidden` (bool hidden)
- void `set_identity`
- void `set_ignore_transform_notification` (bool enabled)
- void `set_notify_local_transform` (bool enable)
- void `set_rotation` (Vector3 rotation_rad)
• void set_rotation_deg (Vector3 rotation_deg)
• void set_scale (Vector3 scale)
• void set_transform (Transform local)
Set the transform locally, relative to the parent spatial node.
• void set_translation (Vector3 translation)
• void show ()
• void translate (Vector3 offset)
• void update_gizmo (

12.294 SpatialGizmo

Inherits: Reference < Object
Inherited By: EditorSpatialGizmo
Category: Core

12.294.1 Brief Description

12.295 SpatialPlayer

Inherits: Spatial < Node < Object
Inherited By: SpatialStreamPlayer, SpatialSamplePlayer
Category: Core

12.295.1 Brief Description

12.295.2 Member Functions

<table>
<thead>
<tr>
<th>float</th>
<th>get_param (int param) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>set_param (int param, float value)</td>
</tr>
</tbody>
</table>

12.295.3 Numeric Constants

• PARAM_VOLUME_DB = 0
• PARAM_PITCH_SCALE = 1
• PARAM_ATTENUATION_MIN_DISTANCE = 2
• PARAM_ATTENUATION_MAX_DISTANCE = 3
• PARAM_ATTENUATION_DISTANCE_EXP = 4
• PARAM_EMISSION_CONE_DEGREES = 5
• PARAM_EMISSION_CONE_ATTENUATION_DB = 6
• PARAM_MAX = 7

### 12.295.4 Member Function Description

- `float get_param ( int param ) const`
- `void set_param ( int param, float value )`

### 12.296 SpatialSamplePlayer

Inherits: SpatialPlayer < Spatial < Node < Object

Category: Core

#### 12.296.1 Brief Description

#### 12.296.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td><code>get_polyphony ( ) const</code></td>
</tr>
<tr>
<td>SampleLibrary</td>
<td><code>get_sample_library ( ) const</code></td>
</tr>
<tr>
<td>bool</td>
<td><code>is_voice_active ( int voice ) const</code></td>
</tr>
<tr>
<td>int</td>
<td><code>play ( String sample, int voice=-2 )</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_polyphony ( int voices )</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_sample_library ( SampleLibrary library )</code></td>
</tr>
<tr>
<td>void</td>
<td><code>stop_all ( )</code></td>
</tr>
<tr>
<td>void</td>
<td><code>stop_voice ( int voice )</code></td>
</tr>
<tr>
<td>void</td>
<td><code>voice_set_pitch_scale ( int voice, float ratio )</code></td>
</tr>
<tr>
<td>void</td>
<td><code>voice_set_volume_scale_db ( int voice, float db )</code></td>
</tr>
</tbody>
</table>

#### 12.296.3 Numeric Constants

- `INVALID_VOICE = -1`
- `NEXT_VOICE = -2`

#### 12.296.4 Member Function Description

- `int get_polyphony ( ) const`
- `SampleLibrary get_sample_library ( ) const`
- `bool is_voice_active ( int voice ) const`
- `int play ( String sample, int voice=-2 )`
- `void set_polyphony ( int voices )`
- `void set_sample_library ( SampleLibrary library )`
- `void stop_all ( )`
- `void stop_voice ( int voice )`
- `void voice_set_pitch_scale ( int voice, float ratio )`
void voice_set_volume_scale_db ( int voice, float db )

12.297 SpatialSound2DServer

Inherits: Object
Inherited By: SpatialSound2DServerSW
Category: Core

12.297.1 Brief Description
Server for Spatial 2D Sound.

12.297.2 Description
Server for Spatial 2D Sound.

12.298 SpatialSound2DServerSW

Inherits: SpatialSound2DServer < Object
Category: Core

12.298.1 Brief Description

12.299 SpatialSoundServer

Inherits: Object
Inherited By: SpatialSoundServerSW
Category: Core

12.299.1 Brief Description

12.300 SpatialSoundServerSW

Inherits: SpatialSoundServer < Object
Category: Core
12.300.1 Brief Description

12.301 SpatialStreamPlayer

Inherits: SpatialPlayer < Spatial < Node < Object

Category: Core

12.301.1 Brief Description

12.301.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>get_buffering_msec ( ) const</td>
</tr>
<tr>
<td>float</td>
<td>get_length ( ) const</td>
</tr>
<tr>
<td>int</td>
<td>get_loop_count ( ) const</td>
</tr>
<tr>
<td>float</td>
<td>get_loop_restart_time ( ) const</td>
</tr>
<tr>
<td>float</td>
<td>get_pos ( ) const</td>
</tr>
<tr>
<td>AudioStream</td>
<td>get_stream ( ) const</td>
</tr>
<tr>
<td>String</td>
<td>get_stream_name ( ) const</td>
</tr>
<tr>
<td>float</td>
<td>get_volume ( ) const</td>
</tr>
<tr>
<td>float</td>
<td>get_volume_db ( ) const</td>
</tr>
<tr>
<td>bool</td>
<td>has_autoplay ( ) const</td>
</tr>
<tr>
<td>bool</td>
<td>has_loop ( ) const</td>
</tr>
<tr>
<td>bool</td>
<td>is_paused ( ) const</td>
</tr>
<tr>
<td>bool</td>
<td>is_playing ( ) const</td>
</tr>
<tr>
<td>void</td>
<td>play ( float offset=0 )</td>
</tr>
<tr>
<td>void</td>
<td>seek_pos ( float time )</td>
</tr>
<tr>
<td>void</td>
<td>set_autoplay ( bool enabled )</td>
</tr>
<tr>
<td>void</td>
<td>set_buffering_msec ( int msec )</td>
</tr>
<tr>
<td>void</td>
<td>set_loop ( bool enabled )</td>
</tr>
<tr>
<td>void</td>
<td>set_loop_restart_time ( float secs )</td>
</tr>
<tr>
<td>void</td>
<td>set_paused ( bool paused )</td>
</tr>
<tr>
<td>void</td>
<td>set_stream ( AudioStream stream )</td>
</tr>
<tr>
<td>void</td>
<td>set_volume ( float volume )</td>
</tr>
<tr>
<td>void</td>
<td>set_volume_db ( float db )</td>
</tr>
<tr>
<td>void</td>
<td>stop ( )</td>
</tr>
</tbody>
</table>

12.301.3 Member Function Description

- int get_buffering_msec ( ) const
- float get_length ( ) const
- int get_loop_count ( ) const
- float get_loop_restart_time ( ) const
- float get_pos ( ) const
- AudioStream get_stream ( ) const
- String get_stream_name ( ) const
- float get_volume ( ) const
- float get_volume_db ( ) const
12.302 SphereShape

Inherits: Shape < Resource < Reference < Object

Category: Core

12.302.1 Brief Description

12.302.2 Member Functions

<table>
<thead>
<tr>
<th>float</th>
<th>get_radius ( ) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>set_radius ( float radius )</td>
</tr>
</tbody>
</table>

12.302.3 Member Function Description

- float get_radius ( ) const
- void set_radius ( float radius )

12.303 SpinBox

Inherits: Range < Control < CanvasItem < Node < Object

Category: Core
12.303.1 Brief Description

Numerical input text field.

12.303.2 Member Functions

<table>
<thead>
<tr>
<th>Object</th>
<th>get_line_edit ( )</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>get_prefix ( ) const</td>
</tr>
<tr>
<td>String</td>
<td>get_suffix ( ) const</td>
</tr>
<tr>
<td>bool</td>
<td>is_editable ( ) const</td>
</tr>
<tr>
<td>void</td>
<td>set_editable ( bool editable )</td>
</tr>
<tr>
<td>void</td>
<td>set_prefix ( String prefix )</td>
</tr>
<tr>
<td>void</td>
<td>set_suffix ( String suffix )</td>
</tr>
</tbody>
</table>

12.303.3 Description

SpinBox is a numerical input text field. It allows entering integers and floats.

12.303.4 Member Function Description

- **Object** get_line_edit ( )
- **String** get_prefix ( ) const
- **String** get_suffix ( ) const

Return the specific suffix.

- **bool** is_editable ( ) const

Return if the spinbox is editable.

- void set_editable ( bool editable )

Set whether the spinbox is editable.

- void set_prefix ( String prefix )

Set a prefix.

- void set_suffix ( String suffix )

Set a specific suffix.

12.304 SplitContainer

Inherits: *Container* < *Control* < *CanvasItem* < *Node* < *Object*

Inherited By: *HSplitContainer*, *VSplitContainer*

Category: Core

12.304.1 Brief Description

Container for splitting and adjusting.
12.304.2 Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>int get_dragger_visibility()</code></td>
<td>const</td>
</tr>
<tr>
<td><code>int get_split_offset()</code></td>
<td>const</td>
</tr>
<tr>
<td><code>bool is_collapsed()</code></td>
<td>const</td>
</tr>
<tr>
<td><code>void set_collapsed(bool collapsed)</code></td>
<td></td>
</tr>
<tr>
<td><code>void set_dragger_visibility(int mode)</code></td>
<td></td>
</tr>
<tr>
<td><code>void set_split_offset(int offset)</code></td>
<td></td>
</tr>
</tbody>
</table>

12.304.3 Signals

- `dragged(int offset)` Emmitted when the dragger is gragged by user.

12.304.4 Numeric Constants

- `DRAGGER_VISIBLE = 0` — The split dragger is visible.
- `DRAGGER_HIDDEN = 1` — The split dragger is invisible.
- `DRAGGER_HIDDEN_COLLAPSED = 2` — The split dragger is invisible and collapsed.

12.304.5 Description

Container for splitting two controls vertically or horizontally, with a grabber that allows adjusting the split offset or ratio.

12.304.6 Member Function Description

- `int get_dragger_visibility()` const
  Return visibility of the split dragger(One of DRAGGER_VISIBLE, DRAGGER_HIDDEN or DRAGGER_HIDDEN_COLLAPSED).

  - `int get_split_offset()` const
    Return the split offset.

  - `bool is_collapsed()` const
    Return true if the split is collapsed.

  - `void set_collapsed(bool collapsed)`
    Set if the split must be collapsed.

  - `void set_dragger_visibility(int mode)`
    Set visibility of the split dragger(mode must be one of DRAGGER_VISIBLE, DRAGGER_HIDDEN or DRAGGER_HIDDEN_COLLAPSED).

  - `void set_split_offset(int offset)`
    Set the split offset.
12.305 SpotLight

Inherits: Light < VisualInstance < Spatial < Node < Object
Category: Core

12.305.1 Brief Description

Spotlight Light, such as a reflector spotlight or a lantern.

12.305.2 Description

A SpotLight light is a type of Light node that emits lights in a specific direction, in the shape of a cone. The light is attenuated through the distance and this attenuation can be configured by changing the energy, radius and attenuation parameters of Light. TODO: Image of a spotlight.

12.306 Sprite

Inherits: Node2D < CanvasItem < Node < Object
Category: Core

12.306.1 Brief Description

General purpose Sprite node.
### 12.306.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>get_frame ( ) const</td>
</tr>
<tr>
<td>int</td>
<td>get_hframes ( ) const</td>
</tr>
<tr>
<td>Color</td>
<td>get_modulate ( ) const</td>
</tr>
<tr>
<td>Vector2</td>
<td>get_offset ( ) const</td>
</tr>
<tr>
<td>Rect2</td>
<td>get_region_rect ( ) const</td>
</tr>
<tr>
<td>Texture</td>
<td>get_texture ( ) const</td>
</tr>
<tr>
<td>int</td>
<td>get_vframes ( ) const</td>
</tr>
<tr>
<td>bool</td>
<td>is_centered ( ) const</td>
</tr>
<tr>
<td>bool</td>
<td>is_flipped_h ( ) const</td>
</tr>
<tr>
<td>bool</td>
<td>is_flipped_v ( ) const</td>
</tr>
<tr>
<td>bool</td>
<td>is_region ( ) const</td>
</tr>
<tr>
<td>void</td>
<td>set_centered ( bool centered )</td>
</tr>
<tr>
<td>void</td>
<td>set_flip_h ( bool flip_h )</td>
</tr>
<tr>
<td>void</td>
<td>set_flip_v ( bool flip_v )</td>
</tr>
<tr>
<td>void</td>
<td>set_frame ( int frame )</td>
</tr>
<tr>
<td>void</td>
<td>set_hframes ( int hframes )</td>
</tr>
<tr>
<td>void</td>
<td>set_modulate ( Color modulate )</td>
</tr>
<tr>
<td>void</td>
<td>set_offset ( Vector2 offset )</td>
</tr>
<tr>
<td>void</td>
<td>set_region ( bool enabled )</td>
</tr>
<tr>
<td>void</td>
<td>set_region_rect ( Rect2 rect )</td>
</tr>
<tr>
<td>void</td>
<td>set_texture ( Texture texture )</td>
</tr>
<tr>
<td>void</td>
<td>set_vframes ( int vframes )</td>
</tr>
</tbody>
</table>

### 12.306.3 Signals

- frame_changed ( )
- texture_changed ( )

### 12.306.4 Description

General purpose Sprite node. This Sprite node can show any texture as a sprite. The texture can be used as a spritesheet for animation, or only a region from a bigger texture can referenced, like an atlas.

### 12.306.5 Member Function Description

- **int get_frame ( ) const**
  
  Return the texture frame for a sprite-sheet, works when vframes or hframes are greater than 1.

- **int get_hframes ( ) const**
  
  Return the amount of horizontal frames. See set_hframes.

- **Color get_modulate ( ) const**
  
  Return color modulation for the sprite. All sprite pixels are multiplied by this color.

- **Vector2 get_offset ( ) const**
  
  Return sprite draw offset.

- **Rect2 get_region_rect ( ) const**
Return the region rect to read from.

- \textit{Texture get\_texture ( ) const}

Return the base texture for the sprite.

- \textit{int get_vframes ( ) const}

Return the amount of vertical frames. See \textit{set\_vframes}.

- \textit{bool is\_centered ( ) const}

Return if the sprite is centered at the local origin.

- \textit{bool is\_flipped\_h ( ) const}

Return true if the sprite is flipped horizontally.

- \textit{bool is\_flipped\_v ( ) const}

Return true if the sprite is flipped vertically.

- \textit{bool is\_region ( ) const}

Return if the sprite reads from a region.

- \textit{void set\_centered ( bool centered )}

Set whether the sprite should be centered on the origin.

- \textit{void set\_flip\_h ( bool flip\_h )}

Set true to flip the sprite horizontally.

- \textit{void set\_flip\_v ( bool flip\_v )}

Set true to flip the sprite vertically.

- \textit{void set\_frame ( int frame )}

Set the texture frame for a sprite-sheet, works when vframes or hframes are greater than 1.

- \textit{void set\_hframes ( int hframes )}

Set the amount of horizontal frames and converts the sprite into a sprite-sheet. This is useful for animation.

- \textit{void set\_modulate ( Color modulate )}

Set color modulation for the sprite. All sprite pixels are multiplied by this color. Color may contain rgb values above 1 to achieve a highlight effect.

- \textit{void set\_offset ( Vector2 offset )}

Set the sprite draw offset, useful for setting rotation pivots.

- \textit{void set\_region ( bool enabled )}

Set the sprite as a sub-region of a bigger texture. Useful for texture-atlases.

- \textit{void set\_region\_rect ( Rect2 rect )}

Set the region rect to read from.

- \textit{void set\_texture ( Texture texture )}

Set the base texture for the sprite.

- \textit{void set\_vframes ( int vframes )}

Set the amount of vertical frames and converts the sprite into a sprite-sheet. This is useful for animation.
12.307 Sprite3D

Inherits: SpriteBase3D < GeometryInstance < VisualInstance < Spatial < Node < Object

Category: Core

12.307.1 Brief Description

12.307.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>get_frame() const</td>
</tr>
<tr>
<td>int</td>
<td>get_hframes() const</td>
</tr>
<tr>
<td>Rect2</td>
<td>get_region_rect() const</td>
</tr>
<tr>
<td>Texture</td>
<td>get_texture() const</td>
</tr>
<tr>
<td>int</td>
<td>get_vframes() const</td>
</tr>
<tr>
<td>bool</td>
<td>is_region() const</td>
</tr>
<tr>
<td>void</td>
<td>set_frame( int frame )</td>
</tr>
<tr>
<td>void</td>
<td>set_hframes( int hframes )</td>
</tr>
<tr>
<td>void</td>
<td>set_region( bool enabled  )</td>
</tr>
<tr>
<td>void</td>
<td>set_region_rect( Rect2 rect )</td>
</tr>
<tr>
<td>void</td>
<td>set_texture( Texture texture )</td>
</tr>
<tr>
<td>void</td>
<td>set_vframes( int vframes )</td>
</tr>
</tbody>
</table>

12.307.3 Signals

- frame_changed() const

12.307.4 Member Function Description

- int get_frame() const
- int get_hframes() const
- Rect2 get_region_rect() const
- Texture get_texture() const
- int get_vframes() const
- bool is_region() const
- void set_frame( int frame )
- void set_hframes( int hframes )
- void set_region( bool enabled )
- void set_region_rect( Rect2 rect )
- void set_texture( Texture texture )
- void set_vframes( int vframes )
12.308 SpriteBase3D

Inherits: GeometryInstance < VisualInstance < Spatial < Node < Object

Inherited By: AnimatedSprite3D, Sprite3D

Category: Core

12.308.1 Brief Description

12.308.2 Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int get_alpha_cut_mode () const</td>
<td></td>
</tr>
<tr>
<td>int get_axis () const</td>
<td></td>
</tr>
<tr>
<td>bool get_draw_flag (int flag) const</td>
<td></td>
</tr>
<tr>
<td>Rect2 get_item_rect () const</td>
<td></td>
</tr>
<tr>
<td>Color get_modulate () const</td>
<td></td>
</tr>
<tr>
<td>Vector2 get_offset () const</td>
<td></td>
</tr>
<tr>
<td>float get_opacity () const</td>
<td></td>
</tr>
<tr>
<td>float get_pixel_size () const</td>
<td></td>
</tr>
<tr>
<td>bool is_centered ( ) const</td>
<td></td>
</tr>
<tr>
<td>bool is_flipped_h ( ) const</td>
<td></td>
</tr>
<tr>
<td>bool is_flipped_v ( ) const</td>
<td></td>
</tr>
<tr>
<td>void set_alpha_cut_mode (int mode)</td>
<td></td>
</tr>
<tr>
<td>void set_axis (int axis)</td>
<td></td>
</tr>
<tr>
<td>void set_centered (bool centered)</td>
<td></td>
</tr>
<tr>
<td>void set_draw_flag (int flag, bool enabled)</td>
<td></td>
</tr>
<tr>
<td>void set_flip_h (bool flip_h)</td>
<td></td>
</tr>
<tr>
<td>void set_flip_v (bool flip_v)</td>
<td></td>
</tr>
<tr>
<td>void set_modulate (Color modulate)</td>
<td></td>
</tr>
<tr>
<td>void set_offset (Vector2 offset)</td>
<td></td>
</tr>
<tr>
<td>void set_opacity (float opacity)</td>
<td></td>
</tr>
<tr>
<td>void set_pixel_size (float pixel_size)</td>
<td></td>
</tr>
</tbody>
</table>

12.308.3 Numeric Constants

- FLAG_TRANSPARENT = 0
- FLAG_SHADED = 1
- FLAG_DOUBLE_SIDED = 2
- FLAG_ONTOP = 3
- FLAG_MAX = 4
- ALPHA_CUT_DISABLED = 0
- ALPHA_CUT_DISCARD = 1
- ALPHA_CUT_OPAQUE_PREPASS = 2
12.308.4 Member Function Description

- `int get_alpha_cut_mode() const`
- `int get_axis() const`
- `bool get_draw_flag(int flag) const`
- `Rect2 get_item_rect() const`
- `Color get_modulate() const`
- `Vector2 get_offset() const`
- `float get_opacity() const`
- `float get_pixel_size() const`
- `bool is_centered() const`
- `bool is_flipped_h() const`
- `bool is_flipped_v() const`
- `void set_alpha_cut_mode(int mode)`
- `void set_axis(int axis)`
- `void set_centered(bool centered)`
- `void set_draw_flag(int flag, bool enabled)`
- `void set_flip_h(bool flip_h)`
- `void set_flip_v(bool flip_v)`
- `void set_modulate(Color modulate)`
- `void set_offset(Vector2 offset)`
- `void set_opacity(float opacity)`
- `void set_pixel_size(float pixel_size)`

12.309 SpriteFrames

Inherits: Resource < Reference < Object

Category: Core

12.309.1 Brief Description

Sprite frame library for AnimatedSprite.
12.309.2 Member Functions

<table>
<thead>
<tr>
<th>void</th>
<th>add_animation ( String anim )</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>add_frame ( String anim, Object frame, int atpos=-1 )</td>
</tr>
<tr>
<td>void</td>
<td>clear ( String anim )</td>
</tr>
<tr>
<td>void</td>
<td>clear_all ( )</td>
</tr>
<tr>
<td>bool</td>
<td>get_animation_loop ( String anim ) const</td>
</tr>
<tr>
<td>float</td>
<td>get_animation_speed ( String anim ) const</td>
</tr>
<tr>
<td>Object</td>
<td>get_frame ( String anim, int idx ) const</td>
</tr>
<tr>
<td>int</td>
<td>get_frame_count ( String anim ) const</td>
</tr>
<tr>
<td>bool</td>
<td>has_animation ( String anim ) const</td>
</tr>
<tr>
<td>void</td>
<td>remove_animation ( String anim )</td>
</tr>
<tr>
<td>void</td>
<td>remove_frame ( String anim, int idx )</td>
</tr>
<tr>
<td>void</td>
<td>rename_animation ( String anim, String newname )</td>
</tr>
<tr>
<td>void</td>
<td>set_animation_loop ( String anim, bool loop )</td>
</tr>
<tr>
<td>void</td>
<td>set_animation_speed ( String anim, float speed )</td>
</tr>
<tr>
<td>void</td>
<td>set_frame ( String anim, int idx, Object txt )</td>
</tr>
</tbody>
</table>

12.309.3 Description

Sprite frame library for AnimatedSprite.

12.309.4 Member Function Description

- void add_animation ( String anim )
- void add_frame ( String anim, Object frame, int atpos=-1 )
- void clear ( String anim )
- void clear_all ( )
- bool get_animation_loop ( String anim ) const
- float get_animation_speed ( String anim ) const
- Object get_frame ( String anim, int idx ) const
- int get_frame_count ( String anim ) const
- bool has_animation ( String anim ) const
- void remove_animation ( String anim )
- void remove_frame ( String anim, int idx )
- void rename_animation ( String anim, String newname )
- void set_animation_loop ( String anim, bool loop )
- void set_animation_speed ( String anim, float speed )
- void set_frame ( String anim, int idx, Object txt )
12.310 StaticBody

Inherits: PhysicsBody < CollisionObject < Spatial < Node < Object

Category: Core

12.310.1 Brief Description

Static body for 3D Physics.

12.310.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>float</td>
<td>get_bounce() const</td>
</tr>
<tr>
<td>Vector3</td>
<td>get_constant_angular_velocity() const</td>
</tr>
<tr>
<td>Vector3</td>
<td>get_constant_linear_velocity() const</td>
</tr>
<tr>
<td>float</td>
<td>get_friction() const</td>
</tr>
<tr>
<td>void</td>
<td>set_bounce(float bounce)</td>
</tr>
<tr>
<td>void</td>
<td>set_constant_angular_velocity(Vector3 vel)</td>
</tr>
<tr>
<td>void</td>
<td>set_constant_linear_velocity(Vector3 vel)</td>
</tr>
<tr>
<td>void</td>
<td>set_friction(float friction)</td>
</tr>
</tbody>
</table>

12.310.3 Description

Static body for 3D Physics. A static body is a simple body that is not intended to move. They don’t consume any CPU resources in contrast to a RigidBody3D so they are great for scenario collision.

A static body can also be animated by using simulated motion mode. This is useful for implementing functionalities such as moving platforms. When this mode is active the body can be animated and automatically computes linear and angular velocity to apply in that frame and to influence other bodies.

Alternatively, a constant linear or angular velocity can be set for the static body, so even if it doesn’t move, it affects other bodies as if it was moving (this is useful for simulating conveyor belts or conveyor wheels).

12.310.4 Member Function Description

- **float get_bounce() const**
  Return the body bounciness.

- **Vector3 get_constant_angular_velocity() const**
  Return the constant angular velocity for the body.

- **Vector3 get_constant_linear_velocity() const**
  Return the constant linear velocity for the body.

- **float get_friction() const**
  Return the body friction.

- **void set_bounce(float bounce)**
  Set the body bounciness, from 0 (not bouncy) to 1 (bouncy).

- **void set_constant_angular_velocity(Vector3 vel)**
Set a constant angular velocity for the body. This does not rotate the body, but affects other bodies touching it, as if it was rotating.

- void `set_constant_linear_velocity(Vector3 vel)`

Set a constant linear velocity for the body. This does not move the body, but affects other bodies touching it, as if it was moving.

- void `set_friction(float friction)`

Set the body friction, from 0 (frictionless) to 1 (full friction).

### 12.311 StaticBody2D

**Inherits:** `PhysicsBody2D < CollisionObject2D < Node2D < CanvasItem < Node < Object`

**Category:** Core

#### 12.311.1 Brief Description

Static body for 2D Physics.

#### 12.311.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>float</td>
<td><code>get_bounce()</code> const</td>
<td>Return the body bounciness.</td>
</tr>
<tr>
<td>float</td>
<td><code>get_constant_angular_velocity()</code> const</td>
<td></td>
</tr>
<tr>
<td>Vector2</td>
<td><code>get_constant_linear_velocity()</code> const</td>
<td></td>
</tr>
<tr>
<td>float</td>
<td><code>get_friction()</code> const</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td><code>set_bounce(float bounce)</code></td>
<td></td>
</tr>
<tr>
<td>void</td>
<td><code>set_constant_angular_velocity(float vel)</code></td>
<td></td>
</tr>
<tr>
<td>void</td>
<td><code>set_constant_linear_velocity(Vector2 vel)</code></td>
<td></td>
</tr>
<tr>
<td>void</td>
<td><code>set_friction(float friction)</code></td>
<td></td>
</tr>
</tbody>
</table>

#### 12.311.3 Description

Static body for 2D Physics. A static body is a simple body that is not intended to move. They don’t consume any CPU resources in contrast to a `RigidBody2D` so they are great for scenario collision.

A static body can also be animated by using simulated motion mode. This is useful for implementing functionalities such as moving platforms. When this mode is active the body can be animated and automatically computes linear and angular velocity to apply in that frame and to influence other bodies.

Alternatively, a constant linear or angular velocity can be set for the static body, so even if it doesn’t move, it affects other bodies as if it was moving (this is useful for simulating conveyor belts or conveyor wheels).

#### 12.311.4 Member Function Description

- **float `get_bounce()` const**
  
  Return the body bounciness.

- **float `get_constant_angular_velocity()` const**
Return the constant angular velocity for the body.

- \texttt{Vector2 get\_constant\_linear\_velocity ( ) const}

Return the constant linear velocity for the body.

- \texttt{float get\_friction ( ) const}

Return the body friction.

- \texttt{void set\_bounce ( float bounce )}

Set the body bounciness, from 0 (not bouncy) to 1 (bouncy).

- \texttt{void set\_constant\_angular\_velocity ( float vel )}

Set a constant angular velocity for the body. This does not rotate the body, but affects other bodies touching it, as if it was rotating.

- \texttt{void set\_constant\_linear\_velocity ( Vector2 vel )}

Set a constant linear velocity for the body. This does not move the body, but affects other bodies touching it, as if it was moving.

- \texttt{void set\_friction ( float friction )}

Set the body friction, from 0 (frictionless) to 1 (full friction).

\section*{12.312 StreamPeer}

\textbf{Inherits: Reference < Object}

\textbf{Inherited By: StreamPeerBuffer, StreamPeerSSL, StreamPeerTCP}

\textbf{Category: Core}

\subsection*{12.312.1 Brief Description}

Abstraction and base class for stream-based protocols.

\subsection*{12.312.2 Member Functions}

\begin{tabular}{|c|c|}
\hline
\texttt{int} & \texttt{get\_16 ( )} \\
\texttt{int} & \texttt{get\_32 ( )} \\
\texttt{int} & \texttt{get\_64 ( )} \\
\texttt{int} & \texttt{get\_8 ( )} \\
\texttt{int} & \texttt{get\_available\_bytes ( ) const} \\
\texttt{Array} & \texttt{get\_data ( int bytes )} \\
\texttt{float} & \texttt{get\_double ( )} \\
\texttt{float} & \texttt{get\_float ( )} \\
\texttt{Array} & \texttt{get\_partial\_data ( int bytes )} \\
\texttt{String} & \texttt{get\_string ( int bytes )} \\
\texttt{int} & \texttt{get\_u16 ( )} \\
\texttt{int} & \texttt{get\_u32 ( )} \\
\texttt{int} & \texttt{get\_u64 ( )} \\
\hline
\end{tabular}

Continued on next page
### 12.312.3 Description

StreamPeer is an abstraction and base class for stream-based protocols (such as TCP or Unix Sockets). It provides an API for sending and receiving data through streams as raw data or strings.

### 12.312.4 Member Function Description

- **int get_16 ()**
  
  Get a signed 16 bit value from the stream.

- **int get_32 ()**
  
  Get a signed 32 bit value from the stream.

- **int get_64 ()**
  
  Get a signed 64 bit value from the stream.

- **int get_u8 ()**
  
  Get an unsigned byte from the stream.

- **bool is_big_endian_enabled () const**
  
  Indicates if the stream is using big-endian byte order.

- **void put_16 (int val)**
  
  Put an unsigned 16 bit value into the stream.

- **void put_32 (int val)**
  
  Put an unsigned 32 bit value into the stream.

- **void put_64 (int val)**
  
  Put an unsigned 64 bit value into the stream.

- **void put_8 (int val)**
  
  Put an unsigned byte into the stream.

- **String put_utf8_string (int bytes)**
  
  Put a string into the stream with a specified number of UTF-8 bytes.

- **Variant put_var ()**
  
  Put a variant into the stream.

- **void put_double (float val)**
  
  Put a double-precision float into the stream.

- **void put_float (float val)**
  
  Put a single-precision float into the stream.

- **Array put_partial_data (RawArray data)**
  
  Put partial data into the stream.

- **Array put_partial_data (RawArray data)**
  
  Return the amount of bytes this StreamPeer has available.

- **void set_big_endian (bool enable)**
  
  Set the big-endian byte order for the stream.

## Table 12.35 – continued from previous page

<table>
<thead>
<tr>
<th>Function Type</th>
<th>Function Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>get_u8 ()</td>
</tr>
<tr>
<td>String</td>
<td>get_utf8_string (int bytes)</td>
</tr>
<tr>
<td>Variant</td>
<td>get_var ()</td>
</tr>
<tr>
<td>bool</td>
<td>is_big_endian_enabled (const)</td>
</tr>
<tr>
<td>void</td>
<td>put_16 (int val)</td>
</tr>
<tr>
<td>void</td>
<td>put_32 (int val)</td>
</tr>
<tr>
<td>void</td>
<td>put_64 (int val)</td>
</tr>
<tr>
<td>void</td>
<td>put_8 (int val)</td>
</tr>
<tr>
<td>int</td>
<td>put_data (RawArray data)</td>
</tr>
<tr>
<td>void</td>
<td>put_double (float val)</td>
</tr>
<tr>
<td>void</td>
<td>put_float (float val)</td>
</tr>
<tr>
<td>Array</td>
<td>put_partial_data (RawArray data)</td>
</tr>
<tr>
<td>void</td>
<td>put_u16 (int val)</td>
</tr>
<tr>
<td>void</td>
<td>put_u32 (int val)</td>
</tr>
<tr>
<td>void</td>
<td>put_u64 (int val)</td>
</tr>
<tr>
<td>void</td>
<td>put_u8 (int val)</td>
</tr>
<tr>
<td>void</td>
<td>put_utf8_string (String val)</td>
</tr>
<tr>
<td>void</td>
<td>put_var (Variant val)</td>
</tr>
<tr>
<td>void</td>
<td>set_big_endian (bool enable)</td>
</tr>
</tbody>
</table>
Get a single-precision float from the stream.

\* **Array get_partial_data ( int bytes )**

Return a chunk data with the received bytes. The amount of bytes to be received can be requested in the “bytes” argument. If not enough bytes are available, the function will return how many were actually received. This function returns two values, an Error code, and a data array.

\* **String get_string ( int bytes )**

Get a string with byte-length “bytes” from the stream.

\* **int get_u16 ( )**

Get an unsigned 16 bit value from the stream.

\* **int get_u32 ( )**

Get an unsigned 32 bit value from the stream.

\* **int get_u64 ( )**

Get an unsigned 16 bit value from the stream.

\* **int get_u8 ( )**

Get an unsigned byte from the stream.

\* **String get_utf8_string ( int bytes )**

Get an utf8 string with byte-length “bytes” from the stream (this decodes the string sent as utf8).

\* **Variant get_var ( )**

Get a Variant from the stream.

\* **bool is_big_endian_enabled ( ) const**

Return whether this *StreamPeer* is using big-endian format.

\* **void put_16 ( int val )**

Put a signed 16 bit value into the stream.

\* **void put_32 ( int val )**

Put a signed 32 bit value into the stream.

\* **void put_64 ( int val )**

Put a signed 64 bit value into the stream.

\* **void put_8 ( int val )**

Put a signed byte into the stream.

\* **int put_data ( RawArray data )**

Send a chunk of data through the connection, blocking if necessary until the data is done sending. This function returns an Error code.

\* **void put_double ( float val )**

Put a double-precision float into the stream.

\* **void put_float ( float val )**

Put a single-precision float into the stream.

\* **Array put_partial_data ( RawArray data )**
Send a chunk of data through the connection, if all the data could not be sent at once, only part of it will. This function returns two values, an Error code and an integer, describing how much data was actually sent.

- void **put_u16** ( int val )
  
  Put an unsigned 16 bit value into the stream.

- void **put_u32** ( int val )
  
  Put an unsigned 32 bit value into the stream.

- void **put_u64** ( int val )
  
  Put an unsigned 64 bit value into the stream.

- void **put_u8** ( int val )
  
  Put an unsigned byte into the stream.

- void **put_utf8_string** ( String val )
  
  Put a zero-terminated utf8 string into the stream.

- void **put_var** ( Variant val )
  
  Put a Variant into the stream.

- void **set_big_endian** ( bool enable )
  
  Set this StreamPeer to use big-endian format. Default is false.

### 12.313 StreamPeerBuffer

**Inherits:** StreamPeer < Reference < Object

**Category:** Core

#### 12.313.1 Brief Description

#### 12.313.2 Member Functions

<table>
<thead>
<tr>
<th>void</th>
<th>clear ( )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>duplicate ( ) const</td>
</tr>
<tr>
<td>RawArray</td>
<td>get_data_array ( ) const</td>
</tr>
<tr>
<td>int</td>
<td>get_pos ( ) const</td>
</tr>
<tr>
<td>int</td>
<td>get_size ( ) const</td>
</tr>
<tr>
<td>void</td>
<td>resize ( int size )</td>
</tr>
<tr>
<td>void</td>
<td>seek ( int pos )</td>
</tr>
<tr>
<td>void</td>
<td>set_data_array ( RawArray data )</td>
</tr>
</tbody>
</table>

#### 12.313.3 Member Function Description

- void **clear** ( )

- **Object duplicate** ( ) const

- **RawArray get_data_array** ( ) const

- **int get_pos** ( ) const
• int get_size ( ) const
• void resize ( int size )
• void seek ( int pos )
• void set_data_array ( RawArray data )

12.314 StreamPeerSSL

Inherits: StreamPeer < Reference < Object
Category: Core

12.314.1 Brief Description

SSL Stream peer.

12.314.2 Member Functions

<table>
<thead>
<tr>
<th>Error</th>
<th>accept ( StreamPeer stream )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error</td>
<td>connect ( StreamPeer stream, bool validate_certs=false, String for_hostname=&quot;&quot; )</td>
</tr>
<tr>
<td>void</td>
<td>disconnect ( )</td>
</tr>
<tr>
<td>int</td>
<td>get_status ( ) const</td>
</tr>
</tbody>
</table>

12.314.3 Numeric Constants

• STATUS_DISCONNECTED = 0 — A status representing a StreamPeerSSL that is disconnected.
• STATUS_CONNECTED = 1 — A status representing a StreamPeerSSL that is connected to a host.
• STATUS_ERROR_NO_CERTIFICATE = 2 — An error status that shows the peer did not present a SSL certificate and validation was requested.
• STATUS_ERROR_HOSTNAME_MISMATCH = 3 — An error status that shows a mismatch in the SSL certificate domain presented by the host and the domain requested for validation.

12.314.4 Description

SSL Stream peer. This object can be used to connect to SSL servers.

12.314.5 Member Function Description

• Error accept ( StreamPeer stream )
• Error connect ( StreamPeer stream, bool validate_certs=false, String for_hostname="" )

Connect to a peer using an underlying StreamPeer “stream”, when “validate_certs” is true, StreamPeerSSL will validate that the certificate presented by the peer matches the “for_hostname”.

• void disconnect ( )

Disconnect from host.
• `int get_status ( ) const`
Return the status of the connection, one of STATUS_* enum.

12.315 StreamPeerTCP

Inherits: StreamPeer < Reference < Object
Category: Core

12.315.1 Brief Description
TCP Stream peer.

12.315.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td><code>connect ( String host, int port )</code></td>
</tr>
<tr>
<td>void</td>
<td><code>disconnect ( )</code></td>
</tr>
<tr>
<td>String</td>
<td><code>get_connected_host ( ) const</code></td>
</tr>
<tr>
<td>int</td>
<td><code>get_connected_port ( ) const</code></td>
</tr>
<tr>
<td>int</td>
<td><code>get_status ( ) const</code></td>
</tr>
<tr>
<td>bool</td>
<td><code>is_connected ( ) const</code></td>
</tr>
</tbody>
</table>

12.315.3 Numeric Constants

• STATUS_NONE = 0 — The initial status of the StreamPeerTCP, also the status after a disconnect.
• STATUS_CONNECTING = 1 — A status representing a StreamPeerTCP that is connecting to a host.
• STATUS_CONNECTED = 2 — A status representing a StreamPeerTCP that is connected to a host.
• STATUS_ERROR = 3 — A status representing a StreamPeerTCP in error state.

12.315.4 Description
TCP Stream peer. This object can be used to connect to TCP servers, or also is returned by a tcp server.

12.315.5 Member Function Description

• `int connect ( String host, int port )`
Connect to the specified host:port pair. A hostname will be resolved if valid. Returns OK on success or FAILED on failure.

• `void disconnect ( )`
Disconnect from host.

• `String get_connected_host ( ) const`
Return the IP of this peer.

• `int get_connected_port ( ) const`
Return the port of this peer.

- `int get_status ( ) const`

Return the status of the connection, one of STATUS_* enum.

- `bool is_connected ( ) const`

Return whether this peer is connected. Returns true while connecting and connected.

### 12.316 StreamPlayer

**Inherits:** `Node < Object`

**Category:** Core

#### 12.316.1 Brief Description

Base class for audio stream playback.

#### 12.316.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td><code>get_buffering_msec ( ) const</code></td>
</tr>
<tr>
<td>float</td>
<td><code>get_length ( ) const</code></td>
</tr>
<tr>
<td>int</td>
<td><code>get_loop_count ( ) const</code></td>
</tr>
<tr>
<td>float</td>
<td><code>get_loop_restart_time ( ) const</code></td>
</tr>
<tr>
<td>float</td>
<td><code>get_pos ( ) const</code></td>
</tr>
<tr>
<td>AudioStream</td>
<td><code>get_stream ( ) const</code></td>
</tr>
<tr>
<td>String</td>
<td><code>get_stream_name ( ) const</code></td>
</tr>
<tr>
<td>float</td>
<td><code>get_volume ( ) const</code></td>
</tr>
<tr>
<td>float</td>
<td><code>get_volume_db ( ) const</code></td>
</tr>
<tr>
<td>bool</td>
<td><code>has_autoplay ( ) const</code></td>
</tr>
<tr>
<td>bool</td>
<td><code>has_loop ( ) const</code></td>
</tr>
<tr>
<td>bool</td>
<td><code>is_paused ( ) const</code></td>
</tr>
<tr>
<td>bool</td>
<td><code>is_playing ( ) const</code></td>
</tr>
<tr>
<td>void</td>
<td><code>play ( float offset=0 )</code></td>
</tr>
<tr>
<td>void</td>
<td><code>seek_pos ( float time )</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_autoplay ( bool enabled )</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_buffering_msec ( int msec )</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_loop ( bool enabled )</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_loop_restart_time ( float secs )</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_paused ( bool paused )</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_stream ( AudioStream stream )</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_volume ( float volume )</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_volume_db ( float db )</code></td>
</tr>
<tr>
<td>void</td>
<td><code>stop ( )</code></td>
</tr>
</tbody>
</table>

#### 12.316.3 Signals

- `finished ( )`

This signal triggers when the player stops playing. It will not trigger on each loop.
12.316.4 Description

Base class for audio stream playback. Audio stream players inherit from it.

12.316.5 Member Function Description

- `int get_buffering_msec () const`
  Return the size of the audio buffer.

- `float get_length () const`
  Return the length of the stream, in seconds.

- `int get_loop_count () const`
  Return the number of times the playback has looped.

- `float get_loop_restart_time () const`
  Return the point in time the stream will rewind to, when looping.

- `float get_pos () const`
  Return the playback position, in seconds.

- `AudioStream get_stream () const`
  Return the currently assigned stream.

- `String get_stream_name () const`
  Return the name of the currently assigned stream. This is not the file name, but a field inside the file. If no stream is assigned, it returns "<No Stream>".

- `float get_volume () const`
  Return the playback volume for this player.

- `float get_volume_db () const`
  Return the playback volume for this player, in decibels.

- `bool has_autoplay () const`
  Return whether this player will start playing as soon as it enters the scene tree.

- `bool has_loop () const`
  Return whether the stream will be restarted at the end.

- `bool is_paused () const`
  Return whether the playback is currently paused.

- `bool is_playing () const`
  Return whether this player is playing.

- `void play (float offset=0 )`
  Play the currently assigned stream, starting from a given position (in seconds).

- `void seek_pos (float time )`
  Set the playback position, in seconds.

- `void set_autoplay (bool enabled )`
Set whether this player will start playing as soon as it enters the scene tree.

- **void set_buffering_msec ( int msec )**

  Set the size (in milliseconds) of the audio buffer. A long audio buffer protects better against slowdowns, but responds worse to changes (in volume, stream played...). A shorter buffer takes less time to respond to changes, but may stutter if the application suffers some slowdown.

  Default is 500 milliseconds.

- **void set_loop ( bool enabled )**

  Set whether the stream will be restarted at the end.

- **void set_loop_restart_time ( float secs )**

  Set the point in time the stream will rewind to, when looping.

- **void set_paused ( bool paused )**

  Pause stream playback.

- **void set_stream ( AudioStream stream )**

  Set the EventStream this player will play.

- **void set_volume ( float volume )**

  Set the playback volume for this player. This is a float between 0.0 (silent) and 1.0 (full volume). Values over 1.0 will amplify sound even more, but may introduce distortion. Negative values will just invert the output waveform, which produces no audible difference.

- **void set_volume_db ( float db )**

  Set the playback volume for this player, in decibels. This is a float between -80.0 (silent) and 0.0 (full volume). Values under -79.0 get truncated to -80, but values over 0.0 do not, so the warnings for overamplifying (see set_volume) still apply.

- **void stop ( )**

  Stop the playback.

### 12.317 String

**Category:** Built-In Types

#### 12.317.1 Brief Description

Built-in string class.

#### 12.317.2 Member Functions

<table>
<thead>
<tr>
<th>String</th>
<th>String ( bool from )</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>String ( int from )</td>
</tr>
<tr>
<td>String</td>
<td>String ( float from )</td>
</tr>
<tr>
<td>String</td>
<td>String ( Vector2 from )</td>
</tr>
<tr>
<td>String</td>
<td>String ( Rect2 from )</td>
</tr>
</tbody>
</table>

Continued on next page
<table>
<thead>
<tr>
<th>String</th>
<th>String ( Vector3 from )</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>String ( Matrix32 from )</td>
</tr>
<tr>
<td>String</td>
<td>String ( Plane from )</td>
</tr>
<tr>
<td>String</td>
<td>String ( Quat from )</td>
</tr>
<tr>
<td>String</td>
<td>String ( AABB from )</td>
</tr>
<tr>
<td>String</td>
<td>String ( Matrix3 )</td>
</tr>
<tr>
<td>String</td>
<td>String ( Transform from )</td>
</tr>
<tr>
<td>String</td>
<td>String ( Color from )</td>
</tr>
<tr>
<td>String</td>
<td>String ( NodePath from )</td>
</tr>
<tr>
<td>String</td>
<td>String ( RID from )</td>
</tr>
<tr>
<td>String</td>
<td>String ( InputEvent from )</td>
</tr>
<tr>
<td>String</td>
<td>String ( Dictionary from )</td>
</tr>
<tr>
<td>String</td>
<td>String ( Array from )</td>
</tr>
<tr>
<td>String</td>
<td>String ( RawArray from )</td>
</tr>
<tr>
<td>String</td>
<td>String ( IntArray from )</td>
</tr>
<tr>
<td>String</td>
<td>String ( RealArray from )</td>
</tr>
<tr>
<td>String</td>
<td>String ( StringArray from )</td>
</tr>
<tr>
<td>String</td>
<td>String ( Vector2Array from )</td>
</tr>
<tr>
<td>String</td>
<td>String ( Vector3Array from )</td>
</tr>
<tr>
<td>String</td>
<td>String ( ColorArray from )</td>
</tr>
<tr>
<td>String</td>
<td>basename ( )</td>
</tr>
<tr>
<td>bool</td>
<td>begins_with ( String text )</td>
</tr>
<tr>
<td>StringArray</td>
<td>bigrams ( )</td>
</tr>
<tr>
<td>String</td>
<td>c_escape ( )</td>
</tr>
<tr>
<td>String</td>
<td>c_unescape ( )</td>
</tr>
<tr>
<td>String</td>
<td>capitalize ( )</td>
</tr>
<tr>
<td>int</td>
<td>casecmp_to ( String to )</td>
</tr>
<tr>
<td>bool</td>
<td>empty ( )</td>
</tr>
<tr>
<td>bool</td>
<td>ends_with ( String text )</td>
</tr>
<tr>
<td>void</td>
<td>erase ( int pos, int chars )</td>
</tr>
<tr>
<td>String</td>
<td>extension ( )</td>
</tr>
<tr>
<td>int</td>
<td>find ( String what, int from=0 )</td>
</tr>
<tr>
<td>int</td>
<td>find_last ( String what )</td>
</tr>
<tr>
<td>int</td>
<td>findn ( String what, int from=0 )</td>
</tr>
<tr>
<td>String</td>
<td>format ( var values, String placeholder={_} )</td>
</tr>
<tr>
<td>String</td>
<td>get_base_dir ( )</td>
</tr>
<tr>
<td>String</td>
<td>get_file ( )</td>
</tr>
<tr>
<td>int</td>
<td>hash ( )</td>
</tr>
<tr>
<td>int</td>
<td>hex_to_int ( )</td>
</tr>
<tr>
<td>String</td>
<td>insert ( int pos, String what )</td>
</tr>
<tr>
<td>bool</td>
<td>is_abs_path ( )</td>
</tr>
<tr>
<td>bool</td>
<td>is_rel_path ( )</td>
</tr>
<tr>
<td>bool</td>
<td>is_subsequence_of ( String text )</td>
</tr>
<tr>
<td>bool</td>
<td>is_subsequence_ofi ( String text )</td>
</tr>
<tr>
<td>bool</td>
<td>is_valid_float ( )</td>
</tr>
<tr>
<td>bool</td>
<td>is_valid_html_color ( )</td>
</tr>
<tr>
<td>bool</td>
<td>is_valid_identifier ( )</td>
</tr>
<tr>
<td>bool</td>
<td>is_valid_integer ( )</td>
</tr>
<tr>
<td>bool</td>
<td>is_valid_ip_address ( )</td>
</tr>
<tr>
<td>String</td>
<td>json_escape ( )</td>
</tr>
</tbody>
</table>

Continued on next page
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>left (int pos)</td>
</tr>
<tr>
<td>int</td>
<td>length()</td>
</tr>
<tr>
<td>bool</td>
<td>match (String expr)</td>
</tr>
<tr>
<td>bool</td>
<td>matchn (String expr)</td>
</tr>
<tr>
<td>RawArray</td>
<td>md5_buffer()</td>
</tr>
<tr>
<td>String</td>
<td>md5_text()</td>
</tr>
<tr>
<td>int</td>
<td>nocasecmp_to (String to)</td>
</tr>
<tr>
<td>int</td>
<td>ord_at (int at)</td>
</tr>
<tr>
<td>String</td>
<td>pad_decimals (int digits)</td>
</tr>
<tr>
<td>String</td>
<td>pad_zeros (int digits)</td>
</tr>
<tr>
<td>String</td>
<td>percent_decode()</td>
</tr>
<tr>
<td>String</td>
<td>percent_encode()</td>
</tr>
<tr>
<td>String</td>
<td>plus_file (String file)</td>
</tr>
<tr>
<td>String</td>
<td>replace (String what, String forwhat)</td>
</tr>
<tr>
<td>String</td>
<td>replacen (String what, String forwhat)</td>
</tr>
<tr>
<td>int</td>
<td>rfind (String what, int from=-1)</td>
</tr>
<tr>
<td>int</td>
<td>rfindn (String what, int from=-1)</td>
</tr>
<tr>
<td>String</td>
<td>right (int pos)</td>
</tr>
<tr>
<td>RawArray</td>
<td>sha256_buffer()</td>
</tr>
<tr>
<td>String</td>
<td>sha256_text()</td>
</tr>
<tr>
<td>float</td>
<td>similarity (String text)</td>
</tr>
<tr>
<td>StringArray</td>
<td>split (String divisor, bool allow_empty=True)</td>
</tr>
<tr>
<td>RealArray</td>
<td>split_floats (String divisor, bool allow_empty=True)</td>
</tr>
<tr>
<td>String</td>
<td>strip_edges (bool left=True, bool right=True)</td>
</tr>
<tr>
<td>String</td>
<td>substr (int from, int len)</td>
</tr>
<tr>
<td>RawArray</td>
<td>to_ascii()</td>
</tr>
<tr>
<td>float</td>
<td>to_float()</td>
</tr>
<tr>
<td>int</td>
<td>to_int()</td>
</tr>
<tr>
<td>String</td>
<td>to_lower()</td>
</tr>
<tr>
<td>String</td>
<td>to_upper()</td>
</tr>
<tr>
<td>RawArray</td>
<td>to_utf8()</td>
</tr>
<tr>
<td>String</td>
<td>xml_escape()</td>
</tr>
<tr>
<td>String</td>
<td>xml_unescape()</td>
</tr>
</tbody>
</table>

### 12.317.3 Description

This is the built-in string class (and the one used by GDScript). It supports Unicode and provides all necessary means for string handling. Strings are reference counted and use a copy-on-write approach, so passing them around is cheap in resources.

### 12.317.4 Member Function Description

- **String String (bool from)**
- **String String (int from)**
- **String String (float from)**
- **String String (Vector2 from)**
- **String String (Rect2 from)**
- **String String (Vector3 from)**
• **String** String ( *Matrix32* from )
• **String** String ( *Plane* from )
• **String** String ( *Quat* from )
• **String** String ( *AABB* from )
• **String** String ( *Matrix3* from )
• **String** String ( *Transform* from )
• **String** String ( *Color* from )
• **String** String ( *NodePath* from )
• **String** String ( *RID* from )
• **String** String ( *InputEvent* from )
• **String** String ( *Dictionary* from )
• **String** String ( *Array* from )
• **String** String ( *RawArray* from )
• **String** String ( *IntArray* from )
• **String** String ( *RealArray* from )
• **String** String ( *StringArray* from )
• **String** String ( *Vector2Array* from )
• **String** String ( *Vector3Array* from )
• **String** String ( *ColorArray* from )
• **String** basename ( )

If the string is a path to a file, return the path to the file without the extension.

• **bool** begins_with ( **String** text )

Return true if the strings begins with the given string.

• **StringArray** bigrams ( )

Return the bigrams (pairs of consecutive letters) of this string.

• **String** c_escape ( )

Return a copy of the string with special characters escaped using the C language standard.

• **String** c_unescape ( )

Return a copy of the string with escaped characters replaced by their meanings according to the C language standard.

• **String** capitalize ( )

Change the case of some letters. Replace underscores with spaces, convert all letters to lowercase then capitalize first and every letter following the space character. For **capitalize** camelCase mixed_with_underscores it will return Capitalize Camelcase Mixed With Underscores.

• **int** casecmp_to ( **String** to )

Perform a case-sensitive comparison to another string, return -1 if less, 0 if equal and +1 if greater.

• **bool** empty ( )

Return true if the string is empty.
• `bool ends_with ( String text )`
  Return true if the strings ends with the given string.

• `void erase ( int pos, int chars )`
  Erase `chars` characters from the string starting from `pos`.

• `String extension ( )`
  If the string is a path to a file, return the extension.

• `int find ( String what, int from=0 )`
  Find the first occurrence of a substring, return the starting position of the substring or -1 if not found. Optionally, the initial search index can be passed.

• `int find_last ( String what )`
  Find the last occurrence of a substring, return the starting position of the substring or -1 if not found. Optionally, the initial search index can be passed.

• `int findn ( String what, int from=0 )`
  Find the first occurrence of a substring but search as case-insensitive, return the starting position of the substring or -1 if not found. Optionally, the initial search index can be passed.

• `String format ( var values, String placeholder={_} )`

• `String get_base_dir ( )`
  If the string is a path to a file, return the base directory.

• `String get_file ( )`
  If the string is a path to a file, return the file and ignore the base directory.

• `int hash ( )`
  Hash the string and return a 32 bits integer.

• `int hex_to_int ( )`
  Convert a string containing an hexadecimal number into an int.

• `String insert ( int pos, String what )`
  Insert a substring at a given position.

• `bool is_abs_path ( )`
  If the string is a path to a file or directory, return true if the path is absolute.

• `bool is_rel_path ( )`
  If the string is a path to a file or directory, return true if the path is relative.

• `bool is_subsequence_of ( String text )`
  Check whether this string is a subsequence of the given string.

• `bool is_subsequence_ofi ( String text )`
  Check whether this string is a subsequence of the given string, without considering case.

• `bool is_valid_float ( )`
  Check whether the string contains a valid float.

• `bool is_valid_html_color ( )`
Check whether the string contains a valid color in HTML notation.

  • bool is_valid_identifier ( )

Check whether the string is a valid identifier. As is common in programming languages, a valid identifier may contain only letters, digits and underscores (_), and the first character may not be a digit.

  • bool is_valid_integer ( )

Check whether the string contains a valid integer.

  • bool is_valid_ip_address ( )

Check whether the string contains a valid IP address.

  • String json_escape ( )

Return a copy of the string with special characters escaped using the JSON standard.

  • String left ( int pos )

Return an amount of characters from the left of the string.

  • int length ( )

Return the length of the string in characters.

  • bool match ( String expr )

Do a simple expression match, where ‘*’ matches zero or more arbitrary characters and ‘?’ matches any single character except ‘.’.

  • bool matchn ( String expr )

Do a simple case insensitive expression match, using ? and * wildcards (see match).

  • RawArray md5_buffer ( )

Return the MD5 hash of the string as an array of bytes.

  • String md5_text ( )

Return the MD5 hash of the string as a string.

  • int nocasecmp_to ( String to )

Perform a case-insensitive comparison to another string, return -1 if less, 0 if equal and +1 if greater.

  • int ord_at ( int at )

Return the character code at position at.

  • String pad_decimals ( int digits )

Format a number to have an exact number of digits after the decimal point.

  • String pad_zeros ( int digits )

Format a number to have an exact number of digits before the decimal point.

  • String percent_decode ( )

Decode a percent-encoded string. See percent_encode.

  • String percent_encode ( )

Percent-encode a string. This is meant to encode parameters in a URL when sending a HTTP GET request and bodies of form-urlencoded POST request.

  • String plus_file ( String file )
If the string is a path, this concatenates file at the end of the string as a subpath. E.g. "this/is".plus_file("path") == "this/is/path".

- **String replace** (String what, String forwhat)

Replace occurrences of a substring for different ones inside the string.

- **String replacen** (String what, String forwhat)

Replace occurrences of a substring for different ones inside the string, but search case-insensitive.

- **int rfind** (String what, int from=-1)

Perform a search for a substring, but start from the end of the string instead of the beginning.

- **int rfindn** (String what, int from=-1)

Perform a search for a substring, but start from the end of the string instead of the beginning. Also search case-insensitive.

- **String right** (int pos)

Return the right side of the string from a given position.

- **RawArray sha256_buffer** ()

- **String sha256_text** ()

Return the SHA-256 hash of the string as a string.

- **float similarity** (String text)

Return the similarity index of the text compared to this string. 1 means totally similar and 0 means totally dissimilar.

- **StringArray split** (String divisor, bool allow_empty=True)

Split the string by a divisor string, return an array of the substrings. Example “One,Two,Three” will return “One”, “Two”, “Three” if split by “,”.

- **RealArray split_floats** (String divisor, bool allow_empty=True)

Split the string in floats by using a divisor string, return an array of the substrings. Example “1,2.5,3” will return 1,2.5,3 if split by “,”.

- **String strip_edges** (bool left=True, bool right=True)

Return a copy of the string stripped of any non-printable character at the beginning and the end. The optional arguments are used to toggle stripping on the left and right edges respectively.

- **String substr** (int from, int len)

Return part of the string from the position from, with length len.

- **RawArray to_ascii** ()

Convert the String (which is a character array) to RawArray (which is an array of bytes). The conversion is speeded up in comparison to to_utf8() with the assumption that all the characters the String contains are only ASCII characters.

- **float to_float** ()

Convert a string, containing a decimal number, into a float.

- **int to_int** ()

Convert a string, containing an integer number, into an int.

- **String to_lower** ()

Return the string converted to lowercase.
- **String to_upper ( )**
  Return the string converted to uppercase.

- **RawArray to_utf8 ( )**
  Convert the String (which is an array of characters) to RawArray (which is an array of bytes). The conversion is a bit slower than to_ascii(), but supports all UTF-8 characters. Therefore, you should prefer this function over to_ascii().

- **String xml_escape ( )**
  Return a copy of the string with special characters escaped using the XML standard.

- **String xml_unescape ( )**
  Return a copy of the string with escaped characters replaced by their meanings according to the XML standard.

### 12.318 StringArray

**Category:** Built-In Types

#### 12.318.1 Brief Description

String Array.

#### 12.318.2 Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>StringArray ( Array from )</td>
<td>Create from a generic array.</td>
</tr>
<tr>
<td>void append ( String string )</td>
<td>Append an element at the end of the array (alias of push_back).</td>
</tr>
<tr>
<td>void append_array ( StringArray array )</td>
<td></td>
</tr>
<tr>
<td>int insert ( int idx, String string )</td>
<td></td>
</tr>
<tr>
<td>void invert ( )</td>
<td></td>
</tr>
<tr>
<td>void push_back ( String string )</td>
<td></td>
</tr>
<tr>
<td>void remove ( int idx )</td>
<td></td>
</tr>
<tr>
<td>void resize ( int idx )</td>
<td></td>
</tr>
<tr>
<td>void set ( int idx, String string )</td>
<td></td>
</tr>
<tr>
<td>int size ( )</td>
<td></td>
</tr>
</tbody>
</table>

#### 12.318.3 Description

String Array. Array of strings. Can only contain strings. Optimized for memory usage, can’t fragment the memory.

#### 12.318.4 Member Function Description

- **StringArray StringArray ( Array from )**
  Create from a generic array.

- void append ( String string )
  Append an element at the end of the array (alias of push_back).

- void append_array ( StringArray array )
Append an `StringArray` at the end of this array.

- `int insert ( int idx, String string )`
  Insert a new element at a given position in the array. The position must be valid, or at the end of the array (pos==size()).

- `void invert ( )`
  Reverse the order of the elements in the array (so first element will now be the last).

- `void push_back ( String string )`
  Append a string element at end of the array.

- `void remove ( int idx )`
  Remove an element from the array by index.

- `void resize ( int idx )`
  Set the size of the `StringArray`. If larger than the current size it will reserve some space beforehand, and if it is smaller it will cut off the array.

- `void set ( int idx, String string )`
  Change the `String` at the given index.

- `int size ( )`
  Return the size of the array.

### 12.319 StyleBox

**Inherits:** `Resource < Reference < Object`

**Inherited By:** `StyleBoxImageMask, StyleBoxFlat, StyleBoxTexture, StyleBoxEmpty`

**Category:** Core

#### 12.319.1 Brief Description

Base class for drawing stylized boxes for the UI.

#### 12.319.2 Member Functions

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>draw</code></td>
<td><code>draw ( RID canvas_item, Rect2 rect ) const</code></td>
</tr>
<tr>
<td><code>get_center_size</code></td>
<td><code>get_center_size ( ) const</code></td>
</tr>
<tr>
<td><code>get_default_margin</code></td>
<td><code>get_default_margin ( int margin ) const</code></td>
</tr>
<tr>
<td><code>get_margin</code></td>
<td><code>get_margin ( int margin ) const</code></td>
</tr>
<tr>
<td><code>get_minimum_size</code></td>
<td><code>get_minimum_size ( ) const</code></td>
</tr>
<tr>
<td><code>get_offset</code></td>
<td><code>get_offset ( ) const</code></td>
</tr>
<tr>
<td><code>set_default_margin</code></td>
<td><code>set_default_margin ( int margin, float offset )</code></td>
</tr>
<tr>
<td><code>test_mask</code></td>
<td><code>test_mask ( Vector2 point, Rect2 rect ) const</code></td>
</tr>
</tbody>
</table>
12.319.3 Description

StyleBox is Resource that provides an abstract base class for drawing stylized boxes for the UI. StyleBoxes are used for drawing the styles of buttons, line edit backgrounds, tree backgrounds, etc. and also for testing a transparency mask for pointer signals. If mask test fails on a StyleBox assigned as mask to a control, clicks and motion signals will go through it to the one below.

12.319.4 Member Function Description

- **void draw ( RID canvas_item, Rect2 rect ) const**

- **Vector2 get_center_size ( ) const**

- **float get_default_margin ( int margin ) const**

Return the default offset of the margin “margin” (see MARGIN_* enum) of a StyleBox. Controls that draw styleboxes with context inside need to know the margin, so the border of the stylebox is not occluded.

- **float get_margin ( int margin ) const**

Return the offset of margin “margin” (see MARGIN_* enum).

- **Vector2 get_minimum_size ( ) const**

Return the minimum size that this stylebox can be shrunk to.

- **Vector2 get_offset ( ) const**

Return the “offset” of a stylebox. This is a helper function, like writing Vector2(style.get_margin(MARGIN_LEFT),style.get_margin(MARGIN_TOP)).

- **void set_default_margin ( int margin, float offset )**

Set the default offset “offset” of the margin “margin” (see MARGIN_* enum) for a StyleBox. Controls that draw styleboxes with context inside need to know the margin, so the border of the stylebox is not occluded.

- **bool test_mask ( Vector2 point, Rect2 rect ) const**

Test a position in a rectangle, return whether it passes the mask test.

12.320 StyleBoxEmpty

Inherits: StyleBox < Resource < Reference < Object

Category: Core

12.320.1 Brief Description

Empty stylebox (does not display anything).

12.320.2 Description

Empty stylebox (really does not display anything).
12.321 StyleBoxFlat

Inherits: StyleBox < Resource < Reference < Object

Category: Core

12.321.1 Brief Description

Stylebox of a single color.

12.321.2 Member Functions

<table>
<thead>
<tr>
<th>Color</th>
<th>get_bg_color ( ) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool</td>
<td>get_border_blend ( ) const</td>
</tr>
<tr>
<td>int</td>
<td>get_border_size ( ) const</td>
</tr>
<tr>
<td>Color</td>
<td>get_dark_color ( ) const</td>
</tr>
<tr>
<td>bool</td>
<td>get_draw_center ( ) const</td>
</tr>
<tr>
<td>Color</td>
<td>get_light_color ( ) const</td>
</tr>
<tr>
<td>void</td>
<td>set_bg_color ( Color color )</td>
</tr>
<tr>
<td>void</td>
<td>set_border_blend ( bool blend )</td>
</tr>
<tr>
<td>void</td>
<td>set_border_size ( int size )</td>
</tr>
<tr>
<td>void</td>
<td>set_dark_color ( Color color )</td>
</tr>
<tr>
<td>void</td>
<td>set_draw_center ( bool size )</td>
</tr>
<tr>
<td>void</td>
<td>set_light_color ( Color color )</td>
</tr>
</tbody>
</table>

12.321.3 Description

Stylebox of a single color. Displays the stylebox of a single color, alternatively a border with light/dark colors can be assigned.

12.321.4 Member Function Description

- Color get_bg_color ( ) const
- bool get_border_blend ( ) const
- int get_border_size ( ) const
- Color get_dark_color ( ) const
- bool get_draw_center ( ) const
- Color get_light_color ( ) const
- void set_bg_color ( Color color )
- void set_border_blend ( bool blend )
- void set_border_size ( int size )
- void set_dark_color ( Color color )
- void set_draw_center ( bool size )
- void set_light_color ( Color color )
12.322 StyleBoxImageMask

Inherits: StyleBox < Resource < Reference < Object

Category: Core

12.322.1 Brief Description

Image mask based StyleBox, for mask test.

12.322.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool</td>
<td>get_expand ( ) const</td>
</tr>
<tr>
<td>float</td>
<td>get_expand_margin_size ( int margin ) const</td>
</tr>
<tr>
<td>Image</td>
<td>get_image ( ) const</td>
</tr>
<tr>
<td>void</td>
<td>set_expand ( bool expand )</td>
</tr>
<tr>
<td>void</td>
<td>set_expand_margin_size ( int margin, float size )</td>
</tr>
<tr>
<td>void</td>
<td>set_image ( Image image )</td>
</tr>
</tbody>
</table>

12.322.3 Description

This StyleBox is similar to StyleBoxTexture, but only meant to be used for mask testing. It takes an image and applies stretch rules to determine if the point clicked is masked or not.

12.322.4 Member Function Description

- **bool get_expand ( ) const**
  Return whether the expand property is set(default). When expanding, the image will use the same rules as StyleBoxTexture for expand. If not expanding, the image will always be tested at its original size.

- **float get_expand_margin_size ( int margin ) const**
  Return the expand margin size (from enum MARGIN_*). Parts of the image below the size of the margin (and in the direction of the margin) will not expand.

- **Image get_image ( ) const**
  Return the image used for mask testing. (see set_image).

- **void set_expand ( bool expand )**
  Set the expand property (default). When expanding, the image will use the same rules as StyleBoxTexture for expand. If not expanding, the image will always be tested at its original size.

- **void set_expand_margin_size ( int margin, float size )**
  Set an expand margin size (from enum MARGIN_*). Parts of the image below the size of the margin (and in the direction of the margin) will not expand.

- **void set_image ( Image image )**
  Set the image used for mask testing. Pixels (converted to grey) that have a value, less than 0.5 will fail the test.
12.323 StyleBoxTexture

Inherits: StyleBox < Resource < Reference < Object

Category: Core

12.323.1 Brief Description

Texture Based 3x3 scale style.

12.323.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool</td>
<td><code>get_draw_center()</code> const</td>
</tr>
<tr>
<td>float</td>
<td><code>get_expand_margin_size(int margin)</code> const</td>
</tr>
<tr>
<td>float</td>
<td><code>get_margin_size(int margin)</code> const</td>
</tr>
<tr>
<td>Rect2</td>
<td><code>get_region_rect()</code> const</td>
</tr>
<tr>
<td>Texture</td>
<td><code>get_texture()</code> const</td>
</tr>
<tr>
<td>void</td>
<td><code>set_draw_center(bool enable)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_expand_margin_size(int margin, float size)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_margin_size(int margin, float size)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_region_rect(Rect2 region)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_texture(Texture texture)</code></td>
</tr>
</tbody>
</table>

12.323.3 Signals

- `texture_changed()`

12.323.4 Description

Texture Based 3x3 scale style. This stylebox performs a 3x3 scaling of a texture, where only the center cell is fully stretched. This allows for the easy creation of bordered styles.

12.323.5 Member Function Description

- `bool get_draw_center()` const
- `float get_expand_margin_size(int margin)` const
- `float get_margin_size(int margin)` const
- `Rect2 get_region_rect()` const
- `Texture get_texture()` const
- `void set_draw_center(bool enable)`
- `void set_expand_margin_size(int margin, float size)`
- `void set_margin_size(int margin, float size)`
- `void set_region_rect(Rect2 region)`
- `void set_texture(Texture texture)`
12.324 SurfaceTool

Inherits: Reference < Object
Category: Core

12.324.1 Brief Description

Helper tool to create geometry.

12.324.2 Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>void add_bones</td>
<td>(IntArray bones)</td>
</tr>
<tr>
<td>void add_color</td>
<td>(Color color)</td>
</tr>
<tr>
<td>void add_index</td>
<td>(int index)</td>
</tr>
<tr>
<td>void add_normal</td>
<td>(Vector3 normal)</td>
</tr>
<tr>
<td>void add_smooth_group</td>
<td>(bool smooth )</td>
</tr>
<tr>
<td>void add_tangent</td>
<td>(Plane tangent)</td>
</tr>
<tr>
<td>void add_to_format</td>
<td>(int flags)</td>
</tr>
<tr>
<td>void add_triangle_fan</td>
<td>(Vector3Array vertexes, Vector2Array uvs=Vector2Array([]), ColorArray colors=ColorArray([ColorArray]), Vector2Array uv2s=Vector2Array([]), Vector3Array normals=Vector3Array([]), Array tangents=Array() )</td>
</tr>
<tr>
<td>void add_uv</td>
<td>(Vector2 uv)</td>
</tr>
<tr>
<td>void add_uv2</td>
<td>(Vector2 uv2)</td>
</tr>
<tr>
<td>void add_vertex</td>
<td>(Vector3 vertex)</td>
</tr>
<tr>
<td>void add_weights</td>
<td>(RealArray weights)</td>
</tr>
<tr>
<td>void append_from</td>
<td>(Mesh existing, int surface, Transform transform)</td>
</tr>
<tr>
<td>void begin</td>
<td>(int primitive)</td>
</tr>
<tr>
<td>void clear</td>
<td></td>
</tr>
<tr>
<td>Mesh commit</td>
<td>(Mesh existing=Null)</td>
</tr>
<tr>
<td>void create_from</td>
<td>(Mesh existing, int surface)</td>
</tr>
<tr>
<td>void deindex</td>
<td></td>
</tr>
<tr>
<td>void generate_normals</td>
<td></td>
</tr>
<tr>
<td>void generate_tangents</td>
<td></td>
</tr>
<tr>
<td>void index</td>
<td></td>
</tr>
<tr>
<td>void set_material</td>
<td>(Material material)</td>
</tr>
</tbody>
</table>

12.324.3 Description

The SurfaceTool is used to construct a Mesh by specifying vertex attributes individually. It can be used to construct a Mesh from script. All properties except index need to be added before a call to add_vertex. For example adding vertex colors and UVs looks like

```javascript
var st = SurfaceTool.new()
st.begin(Mesh.PRIMITIVE_TRIANGLES)
st.add_color(Color(1, 0, 0))
st.add_uv(Vector2(0, 0))
st.add_vertex(Vector3(0, 0, 0))
```

The SurfaceTool now contains one vertex of a triangle which has a UV coordinate and a specified Color. If another vertex were added without calls to add_uv or add_color then the last values would be used.
It is very important that vertex attributes are passed before the call to `add_vertex`, failure to do this will result in an error when committing the vertex information to a mesh.

### 12.324.4 Member Function Description

- **void add_bones (IntArray bones)**
  Add an array of bones for the next Vertex to use.

- **void add_color (Color color)**
  Specify a `Color` for the next Vertex to use.

- **void add_index (int index)**
  Adds an index to index array if you are using indexed Vertices. Does not need to be called before adding Vertex.

- **void add_normal (Vector3 normal)**
  Specify a normal for the next Vertex to use.

- **void add_smooth_group (bool smooth)**
  Specify whether current Vertex (if using only Vertex arrays) or current index (if also using index arrays) should utilize smooth normals for normal calculation.

- **void add_tangent (Plane tangent)**
  Specify a Tangent for the next Vertex to use.

- **void add_to_format (int flags)**

- **void add_triangle_fan (Vector3Array vertexes, Vector2Array uvs=Vector2Array([]), ColorArray colors=ColorArray([]), Vector2Array uv2s=Vector2Array([]), Vector3Array normals=Vector3Array([]), Array tangents=Array())**
  Insert a triangle fan made of array data into `Mesh` being constructed.

- **void add_uv (Vector2 uv)**
  Specify UV Coordinate for next Vertex to use.

- **void add_uv2 (Vector2 uv2)**
  Specify an optional second set of UV coordinates for next Vertex to use.

- **void add_vertex (Vector3 vertex)**
  Specify position of current Vertex. Should be called after specifying other vertex properties (e.g. Color, UV).

- **void add_weights (RealArray weights)**
  Specify weight value for next Vertex to use.

- **void append_from (Mesh existing, int surface, Transform transform)**

- **void begin (int primitive)**
  Called before adding any Vertices. Takes the primitive type as an argument (e.g. `Mesh.PRIMITIVE_TRIANGLES`).

- **void clear ()**
  Clear all information passed into the surface tool so far.

- **Mesh commit (Mesh existing=NULL)**
  Returns a constructed `Mesh` from current information passed in. If an existing `Mesh` is passed in as an argument, will add an extra surface to the existing `Mesh`. 
• void `create_from` (Mesh existing, int surface)
  
• void `deindex` ()

Removes index array by expanding Vertex array.

• void `generate_normals` ()

Generates normals from Vertices so you do not have to do it manually.

• void `generate_tangents` ()

• void `index` ()

Shrinks Vertex array by creating an index array. Avoids reusing Vertices.

• void `set_material` (Material material)

Sets Material to be used by the Mesh you are constructing.

### 12.325 TabContainer

**Inherits:** Control < CanvasItem < Node < Object

**Category:** Core

#### 12.325.1 Brief Description

Tabbed Container.

#### 12.325.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool</td>
<td><code>are_tabs_visible()</code> const</td>
</tr>
<tr>
<td>int</td>
<td><code>get_current_tab()</code> const</td>
</tr>
<tr>
<td>Control</td>
<td><code>get_current_tab_control()</code> const</td>
</tr>
<tr>
<td>Popup</td>
<td><code>get_popup()</code> const</td>
</tr>
<tr>
<td>int</td>
<td><code>get_previous_tab()</code> const</td>
</tr>
<tr>
<td>int</td>
<td><code>get_tab_align()</code> const</td>
</tr>
<tr>
<td>Control</td>
<td><code>get_tab_control(int idx)</code> const</td>
</tr>
<tr>
<td>int</td>
<td><code>get_tab_count()</code> const</td>
</tr>
<tr>
<td>Texture</td>
<td><code>get_tab_icon(int tab_idx)</code> const</td>
</tr>
<tr>
<td>String</td>
<td><code>get_tab_title(int tab_idx)</code> const</td>
</tr>
<tr>
<td>void</td>
<td><code>set_current_tab(int tab_idx)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_popup(Popup popup)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_tab_align(int align)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_tab_icon(int tab_idx, Texture icon)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_tab_title(int tab_idx, String title)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_tabs_visible(bool visible)</code></td>
</tr>
</tbody>
</table>

#### 12.325.3 Signals

• `pre_popup_pressed()` ()

• `tab_changed(int tab)`
Emitted when a tab gets selected. Same behavior as tab_selected signal for backward compatibility. Note: In Godot v3.0+ this will change to be only emitted when tab gets changed.

- `tab_selected ( int tab )`

  Emitted when a tab is being selected, even if it is the same tab.

### 12.325.4 Description

Tabbed Container. Contains several children controls, but shows only one at the same time. Clicking on the top tabs allows to change the currently visible one.

Children controls of this one automatically.

### 12.325.5 Member Function Description

- `bool are_tabs_visible ( ) const`

  Return whether the tabs should be visible or hidden.

- `int get_current_tab ( ) const`

  Return the current tab index that is being shown.

- `Control get_current_tab_control ( ) const`

  Return the current tab control that is being shown.

- `Popup get_popup ( ) const`

- `int get_previous_tab ( ) const`

  Return the previous tab index that was being shown.

- `int get_tab_align ( ) const`

  Return tab alignment, from the ALIGN_* enum.

- `Control get_tab_control ( int idx ) const`

- `int get_tab_count ( ) const`

  Return the amount of tabs.

- `Texture get_tab_icon ( int tab_idx ) const`

- `String get_tab_title ( int tab_idx ) const`

  Return the title for the tab. Tab titles are by default the children node name, but this can be overridden.

- `void set_current_tab ( int tab_idx )`

  Bring a tab (and the Control it represents) to the front, and hide the rest.

- `void set_popup ( Popup popup )`

- `void set_tab_align ( int align )`

  Set tab alignment, from the ALIGN_* enum. Moves tabs to the left, right or center.

- `void set_tab_icon ( int tab_idx, Texture icon )`

  Set an icon for a tab.

- `void set_tab_title ( int tab_idx, String title )`
Set a title for the tab. Tab titles are by default the children node name, but this can be overridden.

- `void set_tabs_visible ( bool visible )`

Set whether the tabs should be visible or hidden.

## 12.326 Tabs

Inherits: `Control < CanvasItem < Node < Object`

Category: Core

### 12.326.1 Brief Description

Tabs Control.

### 12.326.2 Member Functions

```csharp
void add_tab ( String title="", Texture icon=NULL )
void ensure_tab_visible ( int idx )
int get_current_tab ( ) const
int get_tab_align ( ) const
int get_tab_count ( ) const
Texture get_tab_icon ( int tab_idx ) const
String get_tab_title ( int tab_idx ) const
void remove_tab ( int tab_idx )
void set_current_tab ( int tab_idx )
void set_tab_align ( int align )
void set_tab_icon ( int tab_idx, Texture icon )
void set_tab_title ( int tab_idx, String title )
```

### 12.326.3 Signals

- `right_button_pressed ( int tab )`
- `tab_changed ( int tab )`
- `tab_close ( int tab )`

### 12.326.4 Numeric Constants

- `ALIGN_LEFT = 0`
- `ALIGN_CENTER = 1`
- `ALIGN_RIGHT = 2`
- `CLOSE_BUTTON_SHOW_ACTIVE_ONLY = 1`
- `CLOSE_BUTTON_SHOW_ALWAYS = 2`
- `CLOSE_BUTTON_SHOW_NEVER = 0`
12.326.5 Description

Simple tabs control, similar to `TabContainer` but is only in charge of drawing tabs, not interact with children.

12.326.6 Member Function Description

- void `add_tab ( String title="", Texture icon=NULL )`
- void `ensure_tab_visible ( int idx )`
- int `get_current_tab ( ) const`
- int `get_tab_align ( ) const`
- int `get_tab_count ( ) const`
- Texture `get_tab_icon ( int tab_idx ) const`
- String `get_tab_title ( int tab_idx ) const`
- void `remove_tab ( int tab_idx )`
- void `set_current_tab ( int tab_idx )`
- void `set_tab_align ( int align )`
- void `set_tab_icon ( int tab_idx, Texture icon )`
- void `set_tab_title ( int tab_idx, String title )`

12.327 TCP_Server

Inherits: `Reference < Object`

Category: Core

12.327.1 Brief Description

TCP Server.

12.327.2 Member Functions

<table>
<thead>
<tr>
<th>bool</th>
<th>is_connection_available ( ) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>listen ( int port, String bind_address=&quot;*&quot; )</td>
</tr>
<tr>
<td>void</td>
<td>stop ( )</td>
</tr>
<tr>
<td>Object</td>
<td>take_connection ( )</td>
</tr>
</tbody>
</table>

12.327.3 Description

TCP Server class. Listens to connections on a port and returns a `StreamPeerTCP` when got a connection.
### 12.327.4 Member Function Description

- **bool is_connection_available ( ) const**
  
  Return true if a connection is available for taking.

- **int listen ( int port, String bind_address="*" )**

  Listen on the “port” binding to “bind_address”.
  
  If “bind_address” is set as “*” (default), the server will listen on all available addresses (both IPv4 and IPv6).
  
  If “bind_address” is set as “0.0.0.0” (for IPv4) or “::” (for IPv6), the server will listen on all available addresses matching that IP type.
  
  If “bind_address” is set to any valid address (e.g. “192.168.1.101”, “::1”, etc), the server will only listen on the interface with that addresses (or fail if no interface with the given address exists).

- **void stop ( )**

  Stop listening.

- **Object take_connection ( )**

  If a connection is available, return a StreamPeerTCP with the connection.

### 12.328 TestCube

**Inherits:** GeometryInstance < VisualInstance < Spatial < Node < Object

**Category:** Core

#### 12.328.1 Brief Description

A simple cube used for testing in 3D.

#### 12.328.2 Description

The TestCube is a simple 2x2x2 cube with a basic texture. It can be used as a placeholder, to verify how the lighting looks, to test shaders, or any other task you may need a textured model to test with.

### 12.329 TextEdit

**Inherits:** Control < CanvasItem < Node < Object

**Category:** Core

#### 12.329.1 Brief Description

Multiline text editing control.

#### 12.329.2 Member Functions
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void add_color_region(String begin_key, String end_key, Color color, bool line_only=false)</td>
<td>Add a color region to the text editor.</td>
</tr>
<tr>
<td>void add_keyword_color(String keyword, Color color)</td>
<td>Add a keyword color to the text editor.</td>
</tr>
<tr>
<td>void clear_colors()</td>
<td>Clear all colors from the text editor.</td>
</tr>
<tr>
<td>void clear_undo_history()</td>
<td>Clear the undo history of the text editor.</td>
</tr>
<tr>
<td>void copy()</td>
<td>Copy the selected text.</td>
</tr>
<tr>
<td>bool cursor_get_blink_enabled() const</td>
<td>Get the cursor blink enabled status.</td>
</tr>
<tr>
<td>float cursor_get_blink_speed() const</td>
<td>Get the cursor blink speed.</td>
</tr>
<tr>
<td>int cursor_get_column() const</td>
<td>Get the cursor column.</td>
</tr>
<tr>
<td>int cursor_get_line() const</td>
<td>Get the cursor line.</td>
</tr>
<tr>
<td>bool cursor_is_block_mode() const</td>
<td>Get the cursor block mode.</td>
</tr>
<tr>
<td>void cursor_set_blink_enabled(bool enable)</td>
<td>Set the cursor blink enabled status.</td>
</tr>
<tr>
<td>void cursor_set_blink_speed(float blink_speed)</td>
<td>Set the cursor blink speed.</td>
</tr>
<tr>
<td>void cursor_set_block_mode(bool enable)</td>
<td>Set the cursor block mode.</td>
</tr>
<tr>
<td>void cursor_set_column(int column, bool adjust_viewport=false)</td>
<td>Set the cursor column.</td>
</tr>
<tr>
<td>void cursor_set_line(int line, bool adjust_viewport=false)</td>
<td>Set the cursor line.</td>
</tr>
<tr>
<td>void cut()</td>
<td>Cut the selected text.</td>
</tr>
<tr>
<td>String get_line(int line) const</td>
<td>Get a specific line of text.</td>
</tr>
<tr>
<td>int get_line_count() const</td>
<td>Get the line count.</td>
</tr>
<tr>
<td>PopupMenu get_menu() const</td>
<td>Get the menu.</td>
</tr>
<tr>
<td>int get_selection_from_column() const</td>
<td>Get the selection from column.</td>
</tr>
<tr>
<td>int get_selection_from_line() const</td>
<td>Get the selection from line.</td>
</tr>
<tr>
<td>String get_selection_text() const</td>
<td>Get the selection text.</td>
</tr>
<tr>
<td>int get_selection_to_column() const</td>
<td>Get the selection to column.</td>
</tr>
<tr>
<td>int get_selection_to_line() const</td>
<td>Get the selection to line.</td>
</tr>
<tr>
<td>String get_text()</td>
<td>Get the text.</td>
</tr>
<tr>
<td>String get_word_under_cursor() const</td>
<td>Get the word under cursor.</td>
</tr>
<tr>
<td>void insert_text_at_cursor(String text)</td>
<td>Insert text at the cursor.</td>
</tr>
<tr>
<td>bool is_highlight_all_occurrences_enabled() const</td>
<td>Check if all occurrences are highlighted.</td>
</tr>
<tr>
<td>bool is_selection_active() const</td>
<td>Check if the selection is active.</td>
</tr>
<tr>
<td>bool is_show_line_numbers_enabled() const</td>
<td>Check if line numbers are shown.</td>
</tr>
<tr>
<td>bool is_syntax_coloring_enabled() const</td>
<td>Check if syntax coloring is enabled.</td>
</tr>
<tr>
<td>void menu_option(int arg0)</td>
<td>Execute a menu option.</td>
</tr>
<tr>
<td>void paste()</td>
<td>Paste the clipboard content.</td>
</tr>
<tr>
<td>void redo()</td>
<td>Redo the last undo action.</td>
</tr>
<tr>
<td>IntArray search(String flags, int from_line, int from_column, int to_line) const</td>
<td>Search for text.</td>
</tr>
<tr>
<td>void select(int from_line, int from_column, int to_line, int to_column)</td>
<td>Select a range of text.</td>
</tr>
<tr>
<td>void select_all()</td>
<td>Select all text.</td>
</tr>
<tr>
<td>void set_custom_bg_color(Color color)</td>
<td>Set the custom background color.</td>
</tr>
<tr>
<td>void set_highlight_all_occurrences(bool enable)</td>
<td>Set all occurrences to be highlighted.</td>
</tr>
<tr>
<td>void set_max_chars(int amount)</td>
<td>Set the maximum number of characters.</td>
</tr>
<tr>
<td>void set_readonly(bool enable)</td>
<td>Set the text editor to read-only.</td>
</tr>
<tr>
<td>void set_show_line_numbers(bool enable)</td>
<td>Set line numbers to be shown.</td>
</tr>
<tr>
<td>void set_symbol_color(Color color)</td>
<td>Set the symbol color.</td>
</tr>
<tr>
<td>void set_syntax_coloring(bool enable)</td>
<td>Set syntax coloring to be enabled.</td>
</tr>
<tr>
<td>void set_text(String text)</td>
<td>Set the text content.</td>
</tr>
<tr>
<td>void set_wrap(bool enable)</td>
<td>Set the wrap mode.</td>
</tr>
<tr>
<td>void undo()</td>
<td>Undos the last action.</td>
</tr>
</tbody>
</table>

### 12.329.3 Signals

- **breakpoint_toggled(int row)**
Emitted when a breakpoint is placed via the breakpoint gutter.

- `cursor_changed ( )`

Emitted when the cursor changes.

- `request_completion ( )`
- `text_changed ( )`

Emitted when the text changes.

12.329.4 Numeric Constants

- `SEARCH_MATCH_CASE = 1` — Match case when searching.
- `SEARCH_WHOLE_WORDS = 2` — Match whole words when searching.
- `SEARCH_BACKWARDS = 4` — Search from end to beginning.
- `MENU_CUT = 0`
- `MENU_COPY = 1`
- `MENU_PASTE = 2`
- `MENU_CLEAR = 3`
- `MENU_SELECT_ALL = 4`
- `MENU_UNDO = 5`
- `MENU_MAX = 6`

12.329.5 Description

TextEdit is meant for editing large, multiline text. It also has facilities for editing code, such as syntax highlighting support and multiple levels of undo/redo.

12.329.6 Member Function Description

- `void add_color_region ( String begin_key, String end_key, Color color, bool line_only=false )`

Add color region (given the delimiters) and its colors.

- `void add_keyword_color ( String keyword, Color color )`

Add a keyword and its color.

- `void clear_colors ( )`

Clear all the syntax coloring information.

- `void clear_undo_history ( )`

Clear the undo history.

- `void copy ( )`

Copy the current selection.

- `bool cursor_get_blink_enabled ( ) const`

Gets whether the text editor caret is blinking.
• `float cursor_get_blink_speed ( )` const

Gets the text editor caret blink speed.

• `int cursor_get_column ( )` const

Return the column the editing cursor is at.

• `int cursor_get_line ( )` const

Return the line the editing cursor is at.

• `bool cursor_is_block_mode ( )` const

Gets whether the text editor caret is in block mode.

• `void cursor_set_blink_enabled ( bool enable )`

Set the text editor caret to blink.

• `void cursor_set_blink_speed ( float blink_speed )`

Set the text editor caret blink speed. Cannot be less then or equal to 0.

• `void cursor_set_block_mode ( bool enable )`

Set the text editor caret to block mode.

• `void cursor_set_column ( int column, bool adjust_viewport=false )`

• `void cursor_set_line ( int line, bool adjust_viewport=false )`

• `void cut ( )`

Cut the current selection.

• `String get_line ( int line )` const

Return the text of a specific line.

• `int get_line_count ( )` const

Return the amount of total lines in the text.

• `PopupMenu get_menu ( )` const

• `int get_selection_from_column ( )` const

Return the selection begin column.

• `int get_selection_from_line ( )` const

Return the selection begin line.

• `String get_selection_text ( )` const

Return the text inside the selection.

• `int get_selection_to_column ( )` const

Return the selection end column.

• `int get_selection_to_line ( )` const

Return the selection end line.

• `String get_text ( )`

Return the whole text.

• `String get_word_under_cursor ( )` const
• void **insert_text_at_cursor** ( *String* text )
  Insert a given text at the cursor position.

• **bool** is_highlight_all_occurrences_enabled ( ) const
  Returns true if highlight all occurrences is enabled.

• **bool** is_selection_active ( ) const
  Return true if the selection is active.

• **bool** is_show_line_numbers_enabled ( ) const
  Returns true if line numbers are enabled.

• **bool** is_syntax_coloring_enabled ( ) const
  Return true if the syntax coloring is enabled.

• void **menu_option** ( *int* arg0 )

• void **paste** ( )
  Paste the current selection.

• void **redo** ( )
  Perform redo operation.

• **IntArray** search ( *String* flags, *int* from_line, *int* from_column, *int* to_line ) const
  Perform a search inside the text. Search flags can be specified in the SEARCH_* enum.

• void **select** ( *int* from_line, *int* from_column, *int* to_line, *int* to_column )
  Perform selection, from line/column to line/column.

• void **select_all** ( )
  Select all the text.

• void **set_custom_bg_color** ( *Color* color )
  Set a custom background color. A background color with alpha==0 disables this.

• void **set_highlight_all_occurrences** ( *bool* enable )
  Set to enable highlighting all occurrences of the current selection.

• void **set_max_chars** ( *int* amount )
  Set the maximum amount of characters editable.

• void **set_readonly** ( *bool* enable )
  Set the text editor as read-only. Text can be displayed but not edited.

• void **set_show_line_numbers** ( *bool* enable )
  Set to enable showing line numbers.

• void **set_symbol_color** ( *Color* color )
  Set the color for symbols.

• void **set_syntax_coloring** ( *bool* enable )
  Set to enable the syntax coloring.

• void **set_text** ( *String* text )
Set the entire text.

- void `set_wrap(bool enable)`

Enable text wrapping when it goes beyond the edge of what is visible.

- void `undo()`

Perform undo operation.

### 12.330 Texture

**Inherits:** `Resource < Reference < Object`

**Inherited By:** `RenderTargetTexture, AtlasTexture, ImageTexture, LargeTexture`

**Category:** Core

#### 12.330.1 Brief Description

Texture for 2D and 3D.

#### 12.330.2 Member Functions

<table>
<thead>
<tr>
<th>void</th>
<th><code>draw(RID canvas_item, Vector2 pos, Color modulate=Color(1,1,1,1), bool transpose=false)</code> const</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td><code>draw_rect(RID canvas_item, Rect2 rect, bool tile, Color modulate=Color(1,1,1,1), bool transpose=false)</code> const</td>
</tr>
<tr>
<td>void</td>
<td><code>draw_rect_region(RID canvas_item, Rect2 rect, Rect2 src_rect, Color modulate=Color(1,1,1,1), bool transpose=false)</code> const</td>
</tr>
<tr>
<td>int</td>
<td><code>get_flags()</code> const</td>
</tr>
<tr>
<td>int</td>
<td><code>get_height()</code> const</td>
</tr>
<tr>
<td>RID</td>
<td><code>get_rid()</code> const</td>
</tr>
<tr>
<td>Vec-tor2</td>
<td><code>get_size()</code> const</td>
</tr>
<tr>
<td>int</td>
<td><code>get_width()</code> const</td>
</tr>
<tr>
<td>bool</td>
<td><code>has_alpha()</code> const</td>
</tr>
<tr>
<td>void</td>
<td><code>set_flags(int flags)</code></td>
</tr>
</tbody>
</table>

#### 12.330.3 Numeric Constants

- `FLAG_MIPMAPS = 1` — Generate mipmaps, to enable smooth zooming out of the texture.
- `FLAG_REPEAT = 2` — Repeat (instead of clamp to edge).
- `FLAG_FILTER = 4` — Turn on magnifying filter, to enable smooth zooming in of the texture.
- `FLAG_VIDEO_SURFACE = 4096` — Texture is a video surface.
- `FLAGS_DEFAULT = 7` — Default flags. Generate mipmaps, repeat, and filter are enabled.
- `FLAG_ANISOTROPIC_FILTER = 8`
- `FLAG_CONVERT_TO_LINEAR = 16`
- `FLAG_MIRRORED_REPEAT = 32`
12.330.4 Description

A texture works by registering an image in the video hardware, which then can be used in 3D models or 2D Sprite or GUI Control.

12.330.5 Member Function Description

- void draw ( RID canvas_item, Vector2 pos, Color modulate=Color(1,1,1,1), bool transpose=false ) const
- void draw_rect ( RID canvas_item, Rect2 rect, bool tile, Color modulate=Color(1,1,1,1), bool transpose=false ) const
- void draw_rect_region ( RID canvas_item, Rect2 rect, Rect2 src_rect, Color modulate=Color(1,1,1,1), bool transpose=false ) const
- int get_flags ( ) const
  Return the current texture flags.
- int get_height ( ) const
  Return the texture height.
- RID get_rid ( ) const
  Return the texture RID as used in the VisualServer.
- Vector2 get_size ( ) const
  Return the texture size.
- int get_width ( ) const
  Return the texture width.
- bool has_alpha ( ) const
  Change the texture flags.

12.331 TextureButton

Inherits: BaseButton < Control < CanvasItem < Node < Object

Category: Core

12.331.1 Brief Description

Button that can be themed with textures.
### 12.331.2 Member Functions

<table>
<thead>
<tr>
<th>BitMap</th>
<th>get_click_mask() const</th>
</tr>
</thead>
<tbody>
<tr>
<td>Texture</td>
<td>get_disabled_texture() const</td>
</tr>
<tr>
<td>Texture</td>
<td>get_focused_texture() const</td>
</tr>
<tr>
<td>Texture</td>
<td>get_hover_texture() const</td>
</tr>
<tr>
<td>Color</td>
<td>get_modulate() const</td>
</tr>
<tr>
<td>Texture</td>
<td>get_normal_texture() const</td>
</tr>
<tr>
<td>Texture</td>
<td>get_pressed_texture() const</td>
</tr>
<tr>
<td>int</td>
<td>get_resize_mode() const</td>
</tr>
<tr>
<td>int</td>
<td>get_stretch_mode() const</td>
</tr>
<tr>
<td>Vector2</td>
<td>get_texture_scale() const</td>
</tr>
<tr>
<td>void</td>
<td>set_click_mask(BitMap mask)</td>
</tr>
<tr>
<td>void</td>
<td>set_disabled_texture(Texture texture)</td>
</tr>
<tr>
<td>void</td>
<td>set_focused_texture(Texture texture)</td>
</tr>
<tr>
<td>void</td>
<td>set_hover_texture(Texture texture)</td>
</tr>
<tr>
<td>void</td>
<td>set_modulate(Color color)</td>
</tr>
<tr>
<td>void</td>
<td>set_normal_texture(Texture texture)</td>
</tr>
<tr>
<td>void</td>
<td>set_pressed_texture(Texture texture)</td>
</tr>
<tr>
<td>void</td>
<td>set_resize_mode(int p_mode)</td>
</tr>
<tr>
<td>void</td>
<td>set_stretch_mode(int p_mode)</td>
</tr>
<tr>
<td>void</td>
<td>set_texture_scale(Vector2 scale)</td>
</tr>
</tbody>
</table>

### 12.331.3 Numeric Constants

- RESIZE_SCALE = 0
- RESIZE_STRETCH = 1
- STRETCH_SCALE_ON_EXPAND = 0
- STRETCH_SCALE = 1
- STRETCH_TILE = 2
- STRETCH_KEEP = 3
- STRETCH_KEEP_CENTERED = 4
- STRETCH_KEEP_ASPECT = 5
- STRETCH_KEEP_ASPECT_CENTERED = 6
- STRETCH_KEEP_ASPECT_COVERED = 7

### 12.331.4 Description

Button that can be themed with textures. This is like a regular `Button` but can be themed by assigning textures to it. This button is intended to be easy to theme, however a regular button can expand (that uses styleboxes) and still be better if the interface is expect to have internationalization of texts.

Only the normal texture is required, the others are optional.
12.331.5 Member Function Description

- `BitMap get_click_mask ( ) const`
- `Texture get_disabled_texture ( ) const`
- `Texture get_focused_texture ( ) const`
- `Texture get_hover_texture ( ) const`
- `Color get_modulate ( ) const`
- `Texture get_normal_texture ( ) const`
- `Texture get_pressed_texture ( ) const`
- `int get_resize_mode ( ) const`
- `int get_stretch_mode ( ) const`
- `Vector2 get_texture_scale ( ) const`
- `void set_click_mask ( BitMap mask )`
- `void set_disabled_texture ( Texture texture )`
- `void set_focused_texture ( Texture texture )`
- `void set_hover_texture ( Texture texture )`
- `void set_modulate ( Color color )`
- `void set_normal_texture ( Texture texture )`
- `void set_pressed_texture ( Texture texture )`
- `void set_resize_mode ( int p_mode )`
- `void set_stretch_mode ( int p_mode )`
- `void set_texture_scale ( Vector2 scale )`

12.332 TextureFrame

Inherits: `Control < CanvasItem < Node < Object`

Category: Core

12.332.1 Brief Description

Control Frame that draws a texture.
12.332.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color</td>
<td>get_modulate() const</td>
</tr>
<tr>
<td>int</td>
<td>get_stretch_mode() const</td>
</tr>
<tr>
<td>Object</td>
<td>get_texture() const</td>
</tr>
<tr>
<td>bool</td>
<td>has_expand() const</td>
</tr>
<tr>
<td>void</td>
<td>set_expand(bool enable)</td>
</tr>
<tr>
<td>void</td>
<td>set_modulate(Color modulate)</td>
</tr>
<tr>
<td>void</td>
<td>set_stretch_mode(int stretch_mode)</td>
</tr>
<tr>
<td>void</td>
<td>set_texture(Object texture)</td>
</tr>
</tbody>
</table>

12.332.3 Numeric Constants

- STRETCH_SCALE_ON_EXPAND = 0
- STRETCH_SCALE = 1
- STRETCH_TILE = 2
- STRETCH_KEEP = 3
- STRETCH_KEEP_CENTERED = 4
- STRETCH_KEEP_ASPECT = 5
- STRETCH_KEEP_ASPECT_CENTERED = 6
- STRETCH_KEEP_ASPECT_COVERED = 7

12.332.4 Description

Control frame that simply draws an assigned texture. It can stretch or not. It’s a simple way to just show an image in a UI.

12.332.5 Member Function Description

- Color get_modulate() const
- int get_stretch_mode() const
- Object get_texture() const
- bool has_expand() const
- void set_expand(bool enable)
- void set_modulate(Color modulate)
- void set_stretch_mode(int stretch_mode)
- void set_texture(Object texture)

12.333 TextureProgress

Inherits: Range < Control < CanvasItem < Node < Object

Category: Core
12.333.1 Brief Description

Textured progress bar implementation.

12.333.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>float</td>
<td>get_fill_degrees()</td>
</tr>
<tr>
<td>int</td>
<td>get_fill_mode()</td>
</tr>
<tr>
<td>Object</td>
<td>get_over_texture() const</td>
</tr>
<tr>
<td>Object</td>
<td>get_progress_texture() const</td>
</tr>
<tr>
<td>Vector2</td>
<td>get_radial_center_offset()</td>
</tr>
<tr>
<td>float</td>
<td>get_radial_initial_angle()</td>
</tr>
<tr>
<td>Object</td>
<td>get_under_texture() const</td>
</tr>
<tr>
<td>void</td>
<td>set_fill_degrees(float mode)</td>
</tr>
<tr>
<td>void</td>
<td>set_fill_mode(int mode)</td>
</tr>
<tr>
<td>void</td>
<td>set_over_texture(Object tex)</td>
</tr>
<tr>
<td>void</td>
<td>set_progress_texture(Object tex)</td>
</tr>
<tr>
<td>void</td>
<td>set_radial_center_offset(Vector2 mode)</td>
</tr>
<tr>
<td>void</td>
<td>set_radial_initial_angle(float mode)</td>
</tr>
<tr>
<td>void</td>
<td>set_under_texture(Object tex)</td>
</tr>
</tbody>
</table>

12.333.3 Numeric Constants

- FILL_LEFT_TO_RIGHT = 0
- FILL_RIGHT_TO_LEFT = 1
- FILL_TOP_TO_BOTTOM = 2
- FILL_BOTTOM_TO_TOP = 3
- FILL_CLOCKWISE = 4
- FILL_COUNTER_CLOCKWISE = 5

12.333.4 Description

`ProgressBar` implementation that is easier to theme (by just passing a few textures).

12.333.5 Member Function Description

- float get_fill_degrees()
- int get_fill_mode()
- Object get_over_texture() const
- Object get_progress_texture() const
- Vector2 get_radial_center_offset()
- float get_radial_initial_angle()
- Object get_under_texture() const
- void set_fill_degrees(float mode)
• void set_fill_mode ( int mode )
• void set_over_texture ( Object tex )
• void set_progress_texture ( Object tex )
• void set_radial_center_offset ( Vector2 mode )
• void set_radial_initial_angle ( float mode )
• void set_under_texture ( Object tex )

12.334 Theme

Inherits: Resource < Reference < Object

Category: Core

12.334.1 Brief Description

Theme for controls.
12.334.2 Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void clear_color (String name, String type)</code></td>
<td></td>
</tr>
<tr>
<td><code>void clear_constant (String name, String type)</code></td>
<td></td>
</tr>
<tr>
<td><code>void clear_font (String name, String type)</code></td>
<td></td>
</tr>
<tr>
<td><code>void clear_icon (String name, String type)</code></td>
<td></td>
</tr>
<tr>
<td><code>void clear_stylebox (String name, String type)</code></td>
<td></td>
</tr>
<tr>
<td><code>void copy_default_theme ()</code></td>
<td></td>
</tr>
<tr>
<td><code>Color get_color (String name, String type) const</code></td>
<td></td>
</tr>
<tr>
<td><code>StringArray get_color_list (String type) const</code></td>
<td></td>
</tr>
<tr>
<td><code>int get_constant (String name, String type) const</code></td>
<td></td>
</tr>
<tr>
<td><code>StringArray get_constant_list (String type) const</code></td>
<td></td>
</tr>
<tr>
<td><code>Object get_default_font () const</code></td>
<td></td>
</tr>
<tr>
<td><code>Font get_font (String name, String type) const</code></td>
<td></td>
</tr>
<tr>
<td><code>StringArray get_font_list (String type) const</code></td>
<td></td>
</tr>
<tr>
<td><code>Texture get_icon (String name, String type) const</code></td>
<td></td>
</tr>
<tr>
<td><code>StringArray get_icon_list (String type) const</code></td>
<td></td>
</tr>
<tr>
<td><code>StyleBox get_stylebox (String name, String type) const</code></td>
<td></td>
</tr>
<tr>
<td><code>StringArray get_stylebox_list (String type) const</code></td>
<td></td>
</tr>
<tr>
<td><code>StringArray get_stylebox_types () const</code></td>
<td></td>
</tr>
<tr>
<td><code>StringArray get_type_list (String type) const</code></td>
<td></td>
</tr>
<tr>
<td><code>bool has_color (String name, String type) const</code></td>
<td></td>
</tr>
<tr>
<td><code>bool has_constant (String name, String type) const</code></td>
<td></td>
</tr>
<tr>
<td><code>bool has_font (String name, String type) const</code></td>
<td></td>
</tr>
<tr>
<td><code>bool has_icon (String name, String type) const</code></td>
<td></td>
</tr>
<tr>
<td><code>bool has_stylebox (String name, String type) const</code></td>
<td></td>
</tr>
<tr>
<td><code>void set_color (String name, String type, Color color)</code></td>
<td></td>
</tr>
<tr>
<td><code>void set_constant (String name, String type, int constant)</code></td>
<td></td>
</tr>
<tr>
<td><code>void set_default_font (Object font)</code></td>
<td></td>
</tr>
<tr>
<td><code>void set_font (String name, String type, Font font)</code></td>
<td></td>
</tr>
<tr>
<td><code>void set_icon (String name, String type, Texture texture)</code></td>
<td></td>
</tr>
<tr>
<td><code>void set_stylebox (String name, String type, StyleBox texture)</code></td>
<td></td>
</tr>
</tbody>
</table>

12.334.3 Description

Theme for skinning controls. Controls can be skinned individually, but for complex applications it’s more efficient to just create a global theme that defines everything. This theme can be applied to any `Control`, and it and its children will automatically use it.

Theme resources can be alternatively loaded by writing them in a .theme file, see docs for more info.

12.334.4 Member Function Description

- `void clear_color (String name, String type)`
- `void clear_constant (String name, String type)`
- `void clear_font (String name, String type)`
- `void clear_icon (String name, String type)`
- `void clear_stylebox (String name, String type)`
- `void copy_default_theme ()`
• Color get_color (String name, String type) const
• StringArray get_color_list (String type) const
• int get_constant (String name, String type) const
• StringArray get_constant_list (String type) const
• Object get_default_font () const
• Font get_font (String name, String type) const
• StringArray get_font_list (String type) const
• Texture get_icon (String name, String type) const
• StringArray get_icon_list (String type) const
• StyleBox get_stylebox (String name, String type) const
• StringArray get_stylebox_list (String type) const
• StringArray get_stylebox_types () const
• bool has_color (String name, String type) const
• bool has_constant (String name, String type) const
• bool has_font (String name, String type) const
• bool has_icon (String name, String type) const
• bool has_stylebox (String name, String type) const
• void set_color (String name, String type, Color color)
• void set_constant (String name, String type, int constant)
• void set_default_font (Object font)
• void set_font (String name, String type, Font font)
• void set_icon (String name, String type, Texture texture)
• void set_stylebox (String name, String type, StyleBox texture)

12.335 Thread

Inherits: Reference < Object
Category: Core

12.335.1 Brief Description

A unit of execution in a process.
12.335.2 Member Functions

<table>
<thead>
<tr>
<th>String</th>
<th>get_id ( ) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool</td>
<td>is_active ( ) const</td>
</tr>
<tr>
<td>Error</td>
<td>start ( Object instance, String method, Variant userdata=NULL, int priority=1 )</td>
</tr>
<tr>
<td>Variant</td>
<td>wait_to_finish ( )</td>
</tr>
</tbody>
</table>

12.335.3 Numeric Constants

- PRIORITY_LOW = 0
- PRIORITY_NORMAL = 1
- PRIORITY_HIGH = 2

12.335.4 Description

A unit of execution in a process. Can run methods on Objects simultaneously. The use of synchronization via Mutex, Semaphore is advised if working with shared objects.

12.335.5 Member Function Description

- String get_id ( ) const
  Return the id of the thread, uniquely identifying it among all threads.

- bool is_active ( ) const
  Whether this thread is currently active, an active Thread cannot start work on a new method but can be joined with wait_to_finish.

- Error start ( Object instance, String method, Variant userdata=NULL, int priority=1 )
  Start a new Thread, it will run “method” on object “instance” using “userdata” as an argument and running with “priority”, one of PRIORITY_* enum.
  Returns OK on success, or ERR_CANT_CREATE on failure.

- Variant wait_to_finish ( )
  Joins the Thread and waits for it to finish. Returns what the method called returned.

12.336 TileMap

Inherits: Node2D < CanvasItem < Node < Object

Category: Core

12.336.1 Brief Description

Node for 2D tile-based games.
12.336.2 Member Functions
void clear ()
int get_cell ( int x, int y ) const
Vector2 get_cell_size ( ) const
int get_cellv ( Vector2 pos ) const
bool get_center_x ( ) const
bool get_center_y ( ) const
float get_collision_bounce ( ) const
float get_collision_friction ( ) const
int get_collision_layer ( ) const
bool get_collision_layer_bit ( int bit ) const
int get_collision_mask ( ) const
bool get_collision_mask_bit ( int bit ) const
bool get_collision_use_kinematic ( ) const
Matrix32 get_custom_transform ( ) const
int get_half_offset ( ) const
int get_mode ( ) const
int get_occluder_light_mask ( ) const
int get_quadrant_size ( ) const
int get_tile_origin ( ) const
TileSet get_tileset ( ) const
Array get_used_cells ( ) const
Array get_used_cells_by_id ( int id ) const
Rect2 get_used_rect ( )
bool is_cell_transposed ( int x, int y ) const
bool is_cell_x_flipped ( int x, int y ) const
bool is_cell_y_flipped ( int x, int y ) const
bool is_y_sort_mode_enabled ( ) const
Vector2 map_to_world ( Vector2 mappos, bool ignore_half_ofs=false ) const
void set_cell ( int x, int y, int tile, bool flip_x=false, bool flip_y=false, bool transpose=false )
void set_cell_size ( Vector2 size )
void set_cellv ( Vector2 pos, int tile, bool flip_x=false, bool flip_y=false, bool transpose=false )
void set_center_x ( bool enable )
void set_center_y ( bool enable )
void set_collision_bounce ( float value )
void set_collision_friction ( float value )
void set_collision_layer ( int layer )
void set_collision_layer_bit ( int bit, bool value )
void set_collision_mask ( int mask )
void set_collision_mask_bit ( int bit, bool value )
void set_collision_use_kinematic ( bool use_kinematic )
void set_custom_transform ( Matrix32 custom_transform )
void set_half_offset ( int half_offset )
void set_mode ( int mode )
void set_occluder_light_mask ( int mask )
void set_quadrant_size ( int size )
void set_tile_origin ( int origin )
void set_tileset ( TileSet tileset )
void set_y_sort_mode ( bool enable )
Vector2 world_to_map ( Vector2 worldpos ) const
12.336.3 Signals

- settings_changed ( )

Signal indicating that a tilemap setting has changed.

12.336.4 Numeric Constants

- INVALID_CELL = -1 — Returned when a cell doesn’t exist.
- MODE_SQUARE = 0 — Orthogonal orientation mode.
- MODE_ISOMETRIC = 1 — Isometric orientation mode.
- MODE_CUSTOM = 2 — Custom orientation mode.
- HALF_OFFSET_X = 0 — Half offset on the X coordinate.
- HALF_OFFSET_Y = 1 — Half offset on the Y coordinate.
- HALF_OFFSET_DISABLED = 2 — Half offset disabled.
- TILE_ORIGIN_TOP_LEFT = 0 — Tile origin at its top-left corner.
- TILE_ORIGIN_CENTER = 1 — Tile origin at its center.
- TILE_ORIGIN_BOTTOM_LEFT = 2

12.336.5 Description

Node for 2D tile-based games. Tilemaps use a TileSet which contain a list of tiles (textures, their rect and a collision) and are used to create complex grid-based maps.

To optimize drawing and culling (sort of like GridMap), you can specify a quadrant size, so chunks of the map will be batched together at drawing time.

12.336.6 Member Function Description

- void clear ( )

Clear all cells.

- int get_cell ( int x, int y ) const

Return the tile index of the referenced cell.

- Vector2 get_cell_size ( ) const

Return the cell size.

- int get_cellv ( Vector2 pos ) const

Return the tile index of the cell referenced by a Vector2.

- bool get_center_x ( ) const

Return true if tiles are to be centered in x coordinate (by default this is false and they are drawn from upper left cell corner).

- bool get_center_y ( ) const
Return true if tiles are to be centered in y coordinate (by default this is false and they are drawn from upper left cell corner).

- `float get_collision_bounce ( ) const`
  Return the collision bounce parameter.

- `float get_collision_friction ( ) const`
  Return the collision friction parameter.

- `int get_collision_layer ( ) const`
  Return the collision layer.

- `bool get_collision_layer_bit ( int bit ) const`

- `int get_collision_mask ( ) const`

- `bool get_collision_mask_bit ( int bit ) const`

- `bool get_collision_use_kinematic ( ) const`
  Return whether the tilemap handles collisions as a kinematic body.

- `Matrix32 get_custom_transform ( ) const`
  Return the custom transform matrix.

- `int get_half_offset ( ) const`
  Return the current half offset configuration.

- `int get_mode ( ) const`
  Return the orientation mode.

- `int get_occluder_light_mask ( ) const`

- `int get_quadrant_size ( ) const`
  Return the quadrant size.

- `int get_tile_origin ( ) const`
  Return the tile origin configuration.

- `TileSet get_tileset ( ) const`
  Return the current tileset.

- `Array get_used_cells ( ) const`
  Return an array of all cells containing a tile from the tileset (i.e. a tile index different from -1).

- `Array get_used_cells_by_id ( int id ) const`

- `Rect2 get_used_rect ( )`

- `bool is_cell_transposed ( int x, int y ) const`
  Return whether the referenced cell is transposed, i.e. the X and Y axes are swapped (mirroring with regard to the (1,1) vector).

- `bool is_cell_x_flipped ( int x, int y ) const`
  Return whether the referenced cell is flipped over the X axis.

- `bool is_cell_y_flipped ( int x, int y ) const`
Return whether the referenced cell is flipped over the Y axis.

- `bool is_y_sort_mode_enabled ( ) const`

Return the Y sort mode.

- `Vector2 map_to_world ( Vector2 mappos, bool ignore_half_ofs=false ) const`

Return the absolute world position corresponding to the tilemap (grid-based) coordinates given as an argument. Optionally, the tilemap’s potential half offset can be ignored.

- `void set_cell ( int x, int y, int tile, bool flip_x=false, bool flip_y=false, bool transpose=false )`

Set the tile index for the cell referenced by its grid-based X and Y coordinates. A tile index of -1 clears the cell. Optionally, the tile can also be flipped over the X and Y coordinates or transposed.

- `void set_cell_size ( Vector2 size )`

Set the cell size.

- `void set_cellv ( Vector2 pos, int tile, bool flip_x=false, bool flip_y=false, bool transpose=false )`

Set the tile index for the cell referenced by a Vector2 of grid-based coordinates. A tile index of -1 clears the cell. Optionally, the tile can also be flipped over the X and Y axes or transposed.

- `void set_center_x ( bool enable )`

Set tiles to be centered in x coordinate. (by default this is false and they are drawn from upper left cell corner).

- `void set_center_y ( bool enable )`

Set tiles to be centered in y coordinate. (by default this is false and they are drawn from upper left cell corner).

- `void set_collision_bounce ( float value )`

Set the collision bounce parameter. Allowable values range from 0 to 1.

- `void set_collision_friction ( float value )`

Set the collision friction parameter. Allowable values range from 0 to 1.

- `void set_collision_layer ( int layer )`

Set the collision layer. Layers are referenced by binary indexes, so allowable values to describe the 20 available layers range from 0 to 2^20-1.

- `void set_collision_layer_bit ( int bit, bool value )`

- `void set_collision_mask ( int mask )`

Set the collision masks. Masks are referenced by binary indexes, so allowable values to describe the 20 available masks range from 0 to 2^20-1.

- `void set_collision_mask_bit ( int bit, bool value )`

- `void set_collision_use_kinematic ( bool use_kinematic )`

Set the tilemap to handle collisions as a kinematic body (enabled) or a static body (disabled).

- `void set_custom_transform ( Matrix32 custom_transform )`
Set custom transform matrix, to use in combination with the custom orientation mode.

- void set_half_offset ( int half_offset )

Set an half offset on the X coordinate, Y coordinate, or none (use HALF_OFFSET_* constants as argument).

Half offset sets every other tile off by a half tile size in the specified direction.

- void set_mode ( int mode )

Set the orientation mode as square, isometric or custom (use MODE_* constants as argument).

- void set_occluder_light_mask ( int mask )
- void set_quadrant_size ( int size )

Set the quadrant size, this optimizes drawing by batching chunks of map at draw/cull time.

Allowed values are integers ranging from 1 to 128.

- void set_tile_origin ( int origin )

Set the tile origin to the tile center or its top-left corner (use TILE_ORIGIN_* constants as argument).

- void set_tileset ( TileSet tileset )

Set the current tileset.

- void set_y_sort_mode ( bool enable )

Set the Y sort mode. Enabled Y sort mode means that children of the tilemap will be drawn in the order defined by their Y coordinate.

A tile with a higher Y coordinate will therefore be drawn later, potentially covering up the tile(s) above it if its sprite is higher than its cell size.

- Vector2 world_to_map ( Vector2 worldpos ) const

Return the tilemap (grid-based) coordinates corresponding to the absolute world position given as an argument.

## 12.337 TileSet

**Inherits:** Resource &lt; Reference &lt; Object  

**Category:** Core

### 12.337.1 Brief Description

Tile library for tilemaps.
### 12.337.2 Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void clear ( )</td>
<td>Clear all tiles.</td>
</tr>
<tr>
<td>void create_tile ( int id )</td>
<td>Create a new tile which will be referenced by the given ID.</td>
</tr>
<tr>
<td>int find_tile_by_name ( String name ) const</td>
<td>Find the first tile matching the given name.</td>
</tr>
<tr>
<td>int get_last_unused_tile_id ( ) const</td>
<td></td>
</tr>
<tr>
<td>Array get_tiles_ids ( ) const</td>
<td></td>
</tr>
<tr>
<td>void remove_tile ( int id )</td>
<td></td>
</tr>
<tr>
<td>OccluderPolygon2D tile_get_light_occluder ( int id ) const</td>
<td></td>
</tr>
<tr>
<td>CanvasItemMaterial tile_get_material ( int id ) const</td>
<td></td>
</tr>
<tr>
<td>String tile_get_name ( int id ) const</td>
<td></td>
</tr>
<tr>
<td>NavigationPolygon tile_get_navigation_polygon ( int id ) const</td>
<td></td>
</tr>
<tr>
<td>Vector2 tile_get_navigation_polygon_offset ( int id ) const</td>
<td></td>
</tr>
<tr>
<td>Rect2 tile_get_region ( int id ) const</td>
<td></td>
</tr>
<tr>
<td>Shape2D tile_get_shape ( int id ) const</td>
<td></td>
</tr>
<tr>
<td>Vector2 tile_get_shape_offset ( int id ) const</td>
<td></td>
</tr>
<tr>
<td>Array tile_get_shapes ( int id ) const</td>
<td></td>
</tr>
<tr>
<td>Texture tile_get_texture ( int id ) const</td>
<td></td>
</tr>
<tr>
<td>Vector2 tile_get_texture_offset ( int id ) const</td>
<td></td>
</tr>
<tr>
<td>void tile_set_light_occluder ( int id, OccluderPolygon2D light_occluder )</td>
<td></td>
</tr>
<tr>
<td>void tile_set_material ( int id, CanvasItemMaterial material )</td>
<td></td>
</tr>
<tr>
<td>void tile_set_name ( int id, String name )</td>
<td></td>
</tr>
<tr>
<td>void tile_set_navigation_polygon ( int id, NavigationPolygon navigation_polygon )</td>
<td></td>
</tr>
<tr>
<td>void tile_set_navigation_polygon_offset ( int id, Vector2 navigation_polygon_offset )</td>
<td></td>
</tr>
<tr>
<td>void tile_set_occluder_offset ( int id, Vector2 occluder_offset )</td>
<td></td>
</tr>
<tr>
<td>void tile_set_region ( int id, Rect2 region )</td>
<td></td>
</tr>
<tr>
<td>void tile_set_shape ( int id, Shape2D shape )</td>
<td></td>
</tr>
<tr>
<td>void tile_set_shape_offset ( int id, Vector2 shape_offset )</td>
<td></td>
</tr>
<tr>
<td>void tile_set_shapes ( int id, Array shapes )</td>
<td></td>
</tr>
<tr>
<td>void tile_set_texture ( int id, Texture texture )</td>
<td></td>
</tr>
<tr>
<td>void tile_set_texture_offset ( int id, Vector2 texture_offset )</td>
<td></td>
</tr>
</tbody>
</table>

### 12.337.3 Description

A TileSet is a library of tiles for a TileMap. It contains a list of tiles, each consisting of a sprite and optional collision shapes.

Tiles are referenced by a unique integer ID.

### 12.337.4 Member Function Description

- void clear ( )
  
  Clear all tiles.

- void create_tile ( int id )
  
  Create a new tile which will be referenced by the given ID.

- int find_tile_by_name ( String name ) const
  
  Find the first tile matching the given name.


- \textit{int} \text{get\_last\_unused\_tile\_id ( ) const}

Return the ID following the last currently used ID, useful when creating a new tile.

- \textit{Array get\_tiles\_ids ( ) const}

Return an array of all currently used tile IDs.

- \textit{void remove\_tile ( int id )}

Remove the tile referenced by the given ID.

- \textit{OccluderPolygon2D tile\_get\_light\_occluder ( int id ) const}

Return the light occluder of the tile.

- \textit{CanvasItemMaterial tile\_get\_material ( int id ) const}

Return the material of the tile.

- \textit{String tile\_get\_name ( int id ) const}

Return the name of the tile.

- \textit{NavigationPolygon tile\_get\_navigation\_polygon ( int id ) const}

Return the navigation polygon of the tile.

- \textit{Vector2 tile\_get\_navigation\_polygon\_offset ( int id ) const}

Return the offset of the tile’s navigation polygon.

- \textit{Vector2 tile\_get\_occluder\_offset ( int id ) const}

Return the offset of the tile’s light occluder.

- \textit{Rect2 tile\_get\_region ( int id ) const}

Return the tile sub-region in the texture.

- \textit{Shape2D tile\_get\_shape ( int id ) const}

Return the shape of the tile.

- \textit{Vector2 tile\_get\_shape\_offset ( int id ) const}

Return the shape offset of the tile.

- \textit{Array tile\_get\_shapes ( int id ) const}

Return the array of shapes of the tile.

- \textit{Texture tile\_get\_texture ( int id ) const}

Return the texture of the tile.

- \textit{Vector2 tile\_get\_texture\_offset ( int id ) const}

Return the texture offset of the tile.

- \textit{void tile\_set\_light\_occluder ( int id, OccluderPolygon2D light\_occluder )}

Set a light occluder for the tile.

- \textit{void tile\_set\_material ( int id, CanvasItemMaterial material )}

Set the material of the tile.

- \textit{void tile\_set\_name ( int id, String name )}

Set the name of the tile, for descriptive purposes.
• void tile_set_navigation_polygon ( int id, NavigationPolygon navigation_polygon )
Set a navigation polygon for the tile.

• void tile_set_navigation_polygon_offset ( int id, Vector2 navigation_polygon_offset )
Set an offset for the tile’s navigation polygon.

• void tile_set_occluder_offset ( int id, Vector2 occluder_offset )
Set an offset for the tile’s light occluder.

• void tile_set_region ( int id, Rect2 region )
Set the tile sub-region in the texture. This is common in texture atlases.

• void tile_set_shape ( int id, Shape2D shape )
Set a shape for the tile, enabling physics to collide with it.

• void tile_set_shape_offset ( int id, Vector2 shape_offset )
Set the shape offset of the tile.

• void tile_set_shapes ( int id, Array shapes )
Set an array of shapes for the tile, enabling physics to collide with it.

• void tile_set_texture ( int id, Texture texture )
Set the texture of the tile.

• void tile_set_texture_offset ( int id, Vector2 texture_offset )
Set the texture offset of the tile.

12.338 Timer

Inherits: Node < Object

Category: Core

12.338.1 Brief Description

A simple Timer node.
12.338.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>float</td>
<td><code>get_time_left()</code> const</td>
</tr>
<tr>
<td>int</td>
<td><code>get_timer_process_mode()</code> const</td>
</tr>
<tr>
<td>float</td>
<td><code>get_wait_time()</code> const</td>
</tr>
<tr>
<td>bool</td>
<td><code>has_autostart()</code> const</td>
</tr>
<tr>
<td>bool</td>
<td><code>is_active()</code> const</td>
</tr>
<tr>
<td>bool</td>
<td><code>is_one_shot()</code> const</td>
</tr>
<tr>
<td>void</td>
<td><code>set_active(bool active)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_autostart(bool enable)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_one_shot(bool enable)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_timer_process_mode(int mode)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>set_wait_time(float time_sec)</code></td>
</tr>
<tr>
<td>void</td>
<td><code>start()</code></td>
</tr>
<tr>
<td>void</td>
<td><code>stop()</code></td>
</tr>
</tbody>
</table>

12.338.3 Signals

- **timeout()**

  Emitted when the time runs out.

12.338.4 Numeric Constants

- **TIMER_PROCESS_FIXED = 0** — Update the timer at fixed intervals (framerate processing).
- **TIMER_PROCESS_IDLE = 1** — Update the timer during the idle time at each frame.

12.338.5 Description

Timer node. This is a simple node that will emit a timeout callback when the timer runs out. It can optionally be set to loop.

12.338.6 Member Function Description

- **float get_time_left() const**

  Return the time left for timeout in seconds if the timer is active, 0 otherwise.

- **int get_timer_process_mode() const**

  Return the timer’s processing mode.

- **float get_wait_time() const**

  Return the wait time in seconds.

- **bool has_autostart() const**

  Return true if set to automatically start when entering the scene.

- **bool is_active() const**

  Return if the timer is active or not.

- **bool is_one_shot() const**
Return true if configured as one-shot.

- **void set_active ( bool active )**
Set whether the timer is active or not. An inactive timer will be paused until it is activated again.

- **void set_autostart ( bool enable )**
Set to automatically start when entering the scene.

- **void set_one_shot ( bool enable )**
Set as one-shot. If enabled, the timer will stop after timeout, otherwise it will automatically restart.

- **void set_timer_process_mode ( int mode )**
Set the timer’s processing mode (fixed or idle, use TIMER_PROCESS_* constants as argument).

- **void set_wait_time ( float time_sec )**
Set wait time in seconds. When the time is over, it will emit the timeout signal.

- **void start ( )**
Start the timer.

- **void stop ( )**
Stop (cancel) the timer.

### 12.339 ToolButton

**Inherits:** Button < BaseButton < Control < CanvasItem < Node < Object  
**Category:** Core

### 12.339.1 Brief Description

### 12.340 TouchScreenButton

**Inherits:** Node2D < CanvasItem < Node < Object  
**Category:** Core
12.340.1 Brief Description

12.340.2 Member Functions

<table>
<thead>
<tr>
<th>String</th>
<th>get_action() const</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>get_bitmask() const</td>
</tr>
<tr>
<td>Object</td>
<td>get_shape() const</td>
</tr>
<tr>
<td>Object</td>
<td>get_texture() const</td>
</tr>
<tr>
<td>Object</td>
<td>get_texture_pressed() const</td>
</tr>
<tr>
<td>int</td>
<td>get_visibility_mode() const</td>
</tr>
<tr>
<td>bool</td>
<td>is_passby_press_enabled() const</td>
</tr>
<tr>
<td>bool</td>
<td>is_pressed() const</td>
</tr>
<tr>
<td>bool</td>
<td>is_shape_centered() const</td>
</tr>
<tr>
<td>bool</td>
<td>is_shape_visible() const</td>
</tr>
<tr>
<td>void</td>
<td>set_action(String action)</td>
</tr>
<tr>
<td>void</td>
<td>set_bitmask(Object bitmask)</td>
</tr>
<tr>
<td>void</td>
<td>set_passby_press(bool enabled)</td>
</tr>
<tr>
<td>void</td>
<td>set_shape(Object shape)</td>
</tr>
<tr>
<td>void</td>
<td>set_shape_centered(bool bool)</td>
</tr>
<tr>
<td>void</td>
<td>set_shape_visible(bool bool)</td>
</tr>
<tr>
<td>void</td>
<td>set_texture(Object texture)</td>
</tr>
<tr>
<td>void</td>
<td>set_texture_pressed(Object texture_pressed)</td>
</tr>
<tr>
<td>void</td>
<td>set_visibility_mode(int mode)</td>
</tr>
</tbody>
</table>

12.340.3 Signals

- pressed()
- released()

12.340.4 Member Function Description

- String get_action() const
- Object get_bitmask() const
- Object get_shape() const
- Object get_texture() const
- Object get_texture_pressed() const
- int get_visibility_mode() const
- bool is_passby_press_enabled() const
- bool is_pressed() const
- bool is_shape_centered() const
- bool is_shape_visible() const
- void set_action(String action)
- void set_bitmask(Object bitmask)
- void set_passby_press(bool enabled)
• void set_shape ( Object shape )
• void set_shape_centered ( bool bool )
• void set_shape_visible ( bool bool )
• void set_texture ( Object texture )
• void set_texture_pressed ( Object texture_pressed )
• void set_visibility_mode ( int mode )

12.341 Transform

Category: Built-In Types

12.341.1 Brief Description

3D Transformation.

12.341.2 Member Functions

<table>
<thead>
<tr>
<th>Transform</th>
<th>Transform ( Vector3 x_axis, Vector3 y_axis, Vector3 z_axis, Vector3 origin )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transform</td>
<td>Transform ( Matrix3 basis, Vector3 origin )</td>
</tr>
<tr>
<td>Transform</td>
<td>Transform ( Matrix32 from )</td>
</tr>
<tr>
<td>Transform</td>
<td>Transform ( Quat from )</td>
</tr>
<tr>
<td>Transform</td>
<td>Transform ( Matrix3 from )</td>
</tr>
<tr>
<td>Transform</td>
<td>affine_inverse ( )</td>
</tr>
<tr>
<td>Transform</td>
<td>inverse ( )</td>
</tr>
<tr>
<td>Transform</td>
<td>looking_at ( Vector3 target, Vector3 up )</td>
</tr>
<tr>
<td>Transform</td>
<td>orthonormalized ( )</td>
</tr>
<tr>
<td>Transform</td>
<td>rotated ( Vector3 axis, float phi )</td>
</tr>
<tr>
<td>Transform</td>
<td>scaled ( Vector3 scale )</td>
</tr>
<tr>
<td>Transform</td>
<td>translated ( Vector3 ofs )</td>
</tr>
<tr>
<td>var</td>
<td>xform ( var v )</td>
</tr>
<tr>
<td>var</td>
<td>xform_inv ( var v )</td>
</tr>
</tbody>
</table>

12.341.3 Member Variables

• Matrix3 basis - The basis contains 3 [Vector3]. X axis, Y axis, and Z axis.
• Vector3 origin - The origin of the transform. Which is the translation offset.

12.341.4 Description

Transform is used to store transformations, including translations. It consists of a Matrix3 “basis” and Vector3 “origin”. Transform is used to represent transformations of any object in space. It is similar to a 4x3 matrix.
12.341.5 Member Function Description

- `Transform Transform ( Vector3 x_axis, Vector3 y_axis, Vector3 z_axis, Vector3 origin )`
  Construct the Transform from four Vector3. Each axis creates the basis.

- `Transform Transform ( Matrix3 basis, Vector3 origin )`
  Construct the Transform from a Matrix3 and Vector3.

- `Transform Transform ( Matrix32 from )`
  Construct the Transform from a Matrix32.

- `Transform Transform ( Quat from )`
  Construct the Transform from a Quat. The origin will be Vector3(0, 0, 0)

- `Transform affine_inverse ( )`
  Returns the inverse of the transform, even if the transform has scale or the axis vectors are not orthogonal.

- `Transform inverse ( )`
  Returns the inverse of the transform.

- `Transform looking_at ( Vector3 target, Vector3 up )`
  Rotate the transform around the up vector to face the target.

- `Transform orthonormalized ( )`
  Returns a transform with the basis orthogonal (90 degrees), and normalized axis vectors.

- `Transform rotated ( Vector3 axis, float phi )`
  Rotate the transform locally.

- `Transform scaled ( Vector3 scale )`
  Scale the transform locally.

- `Transform translated ( Vector3 ofs )`
  Translate the transform locally.

- `var xform ( var v )`
  Transforms vector “v” by this transform.

- `var xform_inv ( var v )`
  Inverse-transforms vector “v” by this transform.

12.342 Translation

Inherits: Resource < Reference < Object

Inherited By: PHashTranslation

Category: Core
12.342.1 Brief Description

Language Translation.

12.342.2 Member Functions

| void | add_message ( String src_message, String xlated_message ) |
| void | erase_message ( String src_message ) |
| String | get_locale ( ) const |
| String | get_message ( String src_message ) const |
| int | get_message_count ( ) const |
| StringArray | get_message_list ( ) const |
| void | set_locale ( String locale ) |

12.342.3 Description

Translations are resources that can be loaded/unloaded on demand. They map a string to another string.

12.342.4 Member Function Description

- void add_message ( String src_message, String xlated_message )
  Add a message for translation.
- void erase_message ( String src_message )
  Erase a message.
- String get_locale ( ) const
  Return the locale of the translation.
- String get_message ( String src_message ) const
  Return a message for translation.
- int get_message_count ( ) const
- StringArray get_message_list ( ) const
  Return all the messages (keys).
- void set_locale ( String locale )
  Set the locale of the translation.

12.343 TranslationServer

Inherits: Object
Category: Core

12.343.1 Brief Description

Server that manages all translations. Translations can be set to it and removed from it.
12.343.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>add_translation (Translation translation)</td>
</tr>
<tr>
<td>void</td>
<td>clear ()</td>
</tr>
<tr>
<td>String</td>
<td>get_locale () const</td>
</tr>
<tr>
<td>void</td>
<td>remove_translation (Translation translation)</td>
</tr>
<tr>
<td>void</td>
<td>set_locale (String locale)</td>
</tr>
<tr>
<td>String</td>
<td>translate (String message) const</td>
</tr>
</tbody>
</table>

12.343.3 Member Function Description

- void add_translation (Translation translation)
- void clear ()
- String get_locale () const
- void remove_translation (Translation translation)
- void set_locale (String locale)
- String translate (String message) const

12.344 Tree

Inherits: Control < CanvasItem < Node < Object
Category: Core

12.344.1 Brief Description

Control to show a tree of items.

12.344.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool</td>
<td>are_column_titles_visible () const</td>
</tr>
<tr>
<td>void</td>
<td>clear ()</td>
</tr>
<tr>
<td>TreeItem</td>
<td>create_item (TreeItem parent=NULL)</td>
</tr>
<tr>
<td>void</td>
<td>ensure_cursor_is_visible ()</td>
</tr>
<tr>
<td>bool</td>
<td>get_allow_rmb_select () const</td>
</tr>
<tr>
<td>int</td>
<td>get_column_at_pos (Vector2 pos) const</td>
</tr>
<tr>
<td>String</td>
<td>get_column_title (int column) const</td>
</tr>
<tr>
<td>int</td>
<td>get_column_width (int column) const</td>
</tr>
<tr>
<td>int</td>
<td>get_columns () const</td>
</tr>
<tr>
<td>Rect2</td>
<td>get_custom_popup_rect () const</td>
</tr>
<tr>
<td>int</td>
<td>get_drop_mode_flags () const</td>
</tr>
<tr>
<td>TreeItem</td>
<td>get_edited () const</td>
</tr>
<tr>
<td>int</td>
<td>get_edited_column () const</td>
</tr>
<tr>
<td>Rect2</td>
<td>get_item_area_rect (TreeItem item, int column=-1) const</td>
</tr>
<tr>
<td>TreeItem</td>
<td>get_item_at_pos (Vector2 pos) const</td>
</tr>
</tbody>
</table>

Continued on next page
Table 12.39 – continued from previous page

<table>
<thead>
<tr>
<th>Function/Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>get_next_selected</strong> (TreeItem from)</td>
<td>Returns the next selected item.</td>
</tr>
<tr>
<td><strong>get_pressed_button</strong> () const</td>
<td>Returns the pressed button.</td>
</tr>
<tr>
<td><strong>get_root</strong> ()</td>
<td>Returns the root of the tree.</td>
</tr>
<tr>
<td><strong>get_scroll</strong> () const</td>
<td>Returns the scroll position.</td>
</tr>
<tr>
<td><strong>get_selected</strong> () const</td>
<td>Returns the selected item.</td>
</tr>
<tr>
<td><strong>get_selected_column</strong> () const</td>
<td>Returns the selected column.</td>
</tr>
<tr>
<td><strong>get_single_select_cell_editing_only_when_already_selected</strong> () const</td>
<td>Determines if single select cell editing is enabled.</td>
</tr>
<tr>
<td><strong>is_folding_hidden</strong> () const</td>
<td>Returns whether folding is hidden.</td>
</tr>
<tr>
<td><strong>set_allow_rmb_select</strong> (bool)</td>
<td>Enables/disables RMB selection.</td>
</tr>
<tr>
<td><strong>set_column_expand</strong> (int column, bool expand)</td>
<td>Expands the specified column.</td>
</tr>
<tr>
<td><strong>set_column_min_width</strong> (int column, int min_width)</td>
<td>Sets the minimum width for the specified column.</td>
</tr>
<tr>
<td><strong>set_column_title</strong> (int column, String title)</td>
<td>Sets the title for the specified column.</td>
</tr>
<tr>
<td><strong>set_column_titles_visible</strong> (bool visible)</td>
<td>Sets whether column titles are visible.</td>
</tr>
<tr>
<td><strong>set_columns</strong> (int amount)</td>
<td>Sets the number of columns.</td>
</tr>
<tr>
<td><strong>set_drop_mode_flags</strong> (int flags)</td>
<td>Sets the drop mode flags.</td>
</tr>
<tr>
<td><strong>set_hide_folding</strong> (bool hide)</td>
<td>Sets whether folding is hidden.</td>
</tr>
<tr>
<td><strong>set_hide_root</strong> (bool enable)</td>
<td>Enables/disables the root node.</td>
</tr>
<tr>
<td><strong>set_select_mode</strong> (int mode)</td>
<td>Sets the selection mode.</td>
</tr>
<tr>
<td><strong>set_single_select_cell_editing_only_when_already_selected</strong> (bool enable)</td>
<td>Enables/disables single select cell editing when already selected.</td>
</tr>
</tbody>
</table>

### 12.344.3 Signals

- **button_pressed** (Object item, int column, int id)
  
  Emitted when a button on the tree was pressed (see `TreeItem.add_button`).

- **cell_selected** ()
  
  Emitted when a cell is selected.

  - **column_title_pressed** (int column)
  
  - **custom_popup_edited** (bool arrow_clicked)
    
    Emitted when a cell with the `CELL_MODE_CUSTOM` is clicked to be edited.

- **empty_tree_rmb_selected** (Vector2 pos)
  
  Emitted when the right mouse button is pressed if RMB selection is active and the tree is empty.

- **item_activated** ()
  
  Emitted when an item is activated (double-clicked).

- **item_collapse** (Object item)
  
  Emitted when an item is collapsed by a click on the folding arrow.

- **item_edited** ()
  
  Emitted when an item is edited.

- **item_rmb_selected** (Vector2 pos)
  
  Emitted when an item is selected with right mouse button.

- **item_selected** ()
  
  Emitted when an item is selected with right mouse button.

- **multi_selected** (Object item, int column, bool selected)
### 12.344.4 Numeric Constants

- `SELECT_SINGLE = 0`
- `SELECT_ROW = 1`
- `SELECT_MULTI = 2`
- `DROP_MODE_DISABLED = 0`
- `DROP_MODE_ON_ITEM = 1`
- `DROP_MODE_INBETWEEN = 2`

### 12.344.5 Description

This shows a tree of items that can be selected, expanded and collapsed. The tree can have multiple columns with custom controls like text editing, buttons and popups. It can be useful for structural displaying and interactions.

Trees are built via code, using `TreeItem` objects to create the structure. They have a single root but multiple root can be simulated if a dummy hidden root is added.

```godot
func _ready():
    var tree = Tree.new()
    var root = tree.create_item()
    tree.set_hide_root(true)
    var child1 = tree.create_item(root)
    var child2 = tree.create_item(root)
    var subchild1 = tree.create_item(child1)
    subchild1.set_text(0, "Subchild1")
```

### 12.344.6 Member Function Description

- **`bool are_column_titles_visible() const`**
  
  Get whether the column titles are being shown.

- **`void clear()`**
  
  Clear the tree. This erases all of the items.

- **`TreeItem create_item(TreeItem parent=NULL)`**
  
  Create an item in the tree and add it as the last child of `parent`. If `parent` is not given, it will be added as the last child of the root, or it’ll be the root itself if the tree is empty.

- **`void ensure_cursor_is_visible()`**
  
  Make the current selected item visible. This will scroll the tree to make sure the selected item is in sight.

- **`bool get_allow_rmb_select() const`**
  
  Get whether a right click can select items.

- **`int get_column_at_pos(Vector2 pos) const`**
  
  Get the column index under the given point.

- **`String get_column_title(int column) const`**
  
  Get the title of the given column.

- **`int get_column_width(int column) const`**
  
  Get the width of the given column.
Get the width of the given column in pixels.
  • int get_columns ( ) const

Get the amount of columns.
  • Rect2 get_custom_popup_rect ( ) const

Get the rectangle for custom popups. Helper to create custom cell controls that display a popup. See TreeItem.set_cell_mode.
  • int get_drop_mode_flags ( ) const

Get the flags of the current drop mode.
  • TreeItem get_edited ( ) const

Get the current edited item. This is only available for custom cell mode.
  • int get_edited_column ( ) const

Get the column of the cell for the current edited icon. This is only available for custom cell mode.
  • Rect2 get_item_area_rect (TreeItem item, int column=-1 ) const

Get the rectangle area of the the specified item. If column is specified, only get the position and size of that column, otherwise get the rectangle containing all columns.
  • TreeItem get_item_at_pos (Vector2 pos ) const

Get the tree item at the specified position (relative to the tree origin position).
  • TreeItem get_next_selected (TreeItem from )

Get the next selected item after the given one.
  • int get_pressed_button ( ) const

Get the index of the last pressed button.
  • TreeItem get_root ( )

Get the root item of the tree.
  • Vector2 get_scroll ( ) const

Get the current scrolling position.
  • TreeItem get_selected ( ) const

Get the currently selected item.
  • int get_selected_column ( ) const

Get the column number of the current selection.
  • bool get_single_select_cell_editing_only_when_already_selected ( ) const

Get whether the editing of a cell should only happen when it is already selected.
  • bool is_folding_hidden ( ) const

Get whether the folding arrow is hidden.
  • void set_allow_rmb_select (bool allow )

Set whether or not a right mouse button click can select items.
  • void set_column_expand (int column, bool expand )

Set whether a column will have the “Expand” flag of Control.
• void set_column_min_width ( int column, int min_width )
Set the minimum width of a column.

• void set_column_title ( int column, String title )
Set the title of a column.

• void set_column_titles_visible ( bool visible )
Set whether the column titles visibility.

• void set_columns ( int amount )
Set the amount of columns.

• void set_drop_mode_flags ( int flags )
Set the drop mode as an OR combination of flags. See DROP_MODE\_\* constants.

• void set_hide_folding ( bool hide )
Set whether the folding arrow should be hidden.

• void set_hide_root ( bool enable )
Set whether the root of the tree should be hidden.

• void set_select_mode ( int mode )
Set the selection mode. Use one of the SELECT\_\* constants.

• void set_single_select_cell_editing_only_when_already_selected ( bool enable )
Set whether the editing of a cell should only happen when it is already selected.

12.345 TreeItem

Inherits: Object
Category: Core

12.345.1 Brief Description

12.345.2 Member Functions

| void | add_button ( int column, Texture button, int button_idx=-1, bool disabled=false, String tooltip="" ) |
| void | clear_custom_bg_color ( int column ) |
| void | clear_custom_color ( int column ) |
| void | deselect ( int column ) |
| void | erase_button ( int column, int button_idx ) |
| Texture | get_button ( int column, int button_idx ) const |
| int | get_button_count ( int column ) const |
| int | get_cell_mode ( int column ) const |
| TreeItem | get_children ( ) |
| Color | get_custom_bg_color ( int column ) const |
| Texture | get_icon ( int column ) const |
| int | get_icon_max_width ( int column ) const |

Continued on next page
Godot Engine Documentation, Release 2.1

Table 12.40 – continued from previous page

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rect2 get_icon_region (int column) const</td>
<td></td>
</tr>
<tr>
<td>void get_metadata (int column) const</td>
<td></td>
</tr>
<tr>
<td>TreeItem get_next ()</td>
<td></td>
</tr>
<tr>
<td>TreeItem get_next_visible ()</td>
<td></td>
</tr>
<tr>
<td>TreeItem get_parent ()</td>
<td></td>
</tr>
<tr>
<td>TreeItem get_prev ()</td>
<td></td>
</tr>
<tr>
<td>TreeItem get_prev_visible ()</td>
<td></td>
</tr>
<tr>
<td>float get_range (int column) const</td>
<td></td>
</tr>
<tr>
<td>Dictionary get_range_config (int column)</td>
<td></td>
</tr>
<tr>
<td>String get_text (int column) const</td>
<td></td>
</tr>
<tr>
<td>String get_tooltip (int column) const</td>
<td></td>
</tr>
<tr>
<td>bool is_button_disabled (int column, int button_idx) const</td>
<td></td>
</tr>
<tr>
<td>bool is_checked (int column) const</td>
<td></td>
</tr>
<tr>
<td>bool is_collapse ()</td>
<td></td>
</tr>
<tr>
<td>bool is_editable (int column)</td>
<td></td>
</tr>
<tr>
<td>bool is_selectable (int column) const</td>
<td></td>
</tr>
<tr>
<td>bool is_selected (int column)</td>
<td></td>
</tr>
<tr>
<td>void move_to_bottom ()</td>
<td></td>
</tr>
<tr>
<td>void move_to_top ()</td>
<td></td>
</tr>
<tr>
<td>TreeItem remove_child (Object child)</td>
<td></td>
</tr>
<tr>
<td>void select (int column)</td>
<td></td>
</tr>
<tr>
<td>void set_button (int column, int button_idx, Texture button)</td>
<td></td>
</tr>
<tr>
<td>void set_cell_mode (int column, int mode)</td>
<td></td>
</tr>
<tr>
<td>void set_checked (int column, bool checked)</td>
<td></td>
</tr>
<tr>
<td>void set_collapse (bool enable)</td>
<td></td>
</tr>
<tr>
<td>void set_custom_bg_color (int column, Color color, bool just_outline=false)</td>
<td></td>
</tr>
<tr>
<td>void set_custom_color (int column, Color color)</td>
<td></td>
</tr>
<tr>
<td>void set_custom_draw (int column, Object object, String callback)</td>
<td></td>
</tr>
<tr>
<td>void set_editable (int column, bool enabled)</td>
<td></td>
</tr>
<tr>
<td>void set_icon (int column, Texture texture)</td>
<td></td>
</tr>
<tr>
<td>void set_icon_max_width (int column, int width)</td>
<td></td>
</tr>
<tr>
<td>void set_icon_region (int column, Rect2 region)</td>
<td></td>
</tr>
<tr>
<td>void set_metadata (int column, Variant meta)</td>
<td></td>
</tr>
<tr>
<td>void set_range (int column, float value)</td>
<td></td>
</tr>
<tr>
<td>void set_range_config (int column, float min, float max, float step, bool expr=false)</td>
<td></td>
</tr>
<tr>
<td>void set_selectable (int column, bool selectable)</td>
<td></td>
</tr>
<tr>
<td>void set_text (int column, String text)</td>
<td></td>
</tr>
<tr>
<td>void set_tooltip (int column, String tooltip)</td>
<td></td>
</tr>
</tbody>
</table>

12.345.3 Numeric Constants

- CELL_MODE_STRING = 0
- CELL_MODE_CHECK = 1
- CELL_MODE_RANGE = 2
- CELL_MODE_RANGE_EXPRESSION = 3
- CELL_MODE_ICON = 4
- CELL_MODE_CUSTOM = 5
12.345.4 Member Function Description

- void add_button ( int column, Texture button, int button_idx=-1, bool disabled=false, String tooltip=""")
- void clear_custom_bg_color ( int column )
- void clear_custom_color ( int column )
- void deselect ( int column )
- void erase_button ( int column, int button_idx )
- Texture get_button ( int column, int button_idx ) const
- int get_button_count ( int column ) const
- int get_cell_mode ( int column ) const
- Treeltem get_children ( )
- Color get_custom_bg_color ( int column ) const
- Texture get_icon ( int column ) const
- int get_icon_max_width ( int column ) const
- Rect2 get_icon_region ( int column ) const
- void get_metadata ( int column ) const
- Treeltem get_next ( )
- Treeltem get_next_visible ( )
- Treeltem get_parent ( )
- Treeltem get_prev ( )
- Treeltem get_prev_visible ( )
- float get_range ( int column ) const
- Dictionary get_range_config ( int column )
- String get_text ( int column ) const
- String get_tooltip ( int column ) const
- bool is_button_disabled ( int column, int button_idx ) const
- bool is_checked ( int column ) const
- bool is_collapsed ( )
- bool is_editable ( int column )
- bool is_selectable ( int column ) const
- bool is_selected ( int column )
- void move_to_bottom ( )
- void move_to_top ( )
- Treeltem remove_child ( Object child )
- void select ( int column )
- void set_button ( int column, int button_idx, Texture button )
- void set_cell_mode ( int column, int mode )
• void set_checked ( int column, bool checked )
• void set_collapsed ( bool enable )
• void set_custom_bg_color ( int column, Color color, bool just_outline=false )
• void set_custom_color ( int column, Color color )
• void set_custom_draw ( int column, Object object, String callback )
• void set_editable ( int column, bool enabled )
• void set_icon ( int column, Texture texture )
• void set_icon_max_width ( int column, int width )
• void set_icon_region ( int column, Rect2 region )
• void set_metadata ( int column, Variant meta )
• void set_range ( int column, float value )
• void set_range_config ( int column, float min, float max, float step, bool expr=false )
• void set_selectable ( int column, bool selectable )
• void set_text ( int column, String text )
• void set_tooltip ( int column, String tooltip )

## 12.346 Tween

**Inherits:** Node < Object  
**Category:** Core

### 12.346.1 Brief Description

Node useful for animations with unknown start and end points.
12.346.2 Member Functions

```cpp
bool follow_method ( Object object, String method, Variant initial_val, Object target, String target_method, float times_in_sec, int trans_type, int ease_type, float delay=0 )

bool follow_property ( Object object, String property, Variant initial_val, Object target, String target_property, float times_in_sec, int trans_type, int ease_type, float delay=0 )

float get_runtime ( ) const

float get_speed ( ) const

int get_tween_process_mode ( ) const

bool interpolate_callback ( Object object, float times_in_sec, String callback, Variant arg1=NULL, Variant arg2=NULL, Variant arg3=NULL, Variant arg4=NULL, Variant arg5=NULL )

bool interpolate_deferred_callback ( Object object, float times_in_sec, String callback, Variant arg1=NULL, Variant arg2=NULL, Variant arg3=NULL, Variant arg4=NULL, Variant arg5=NULL )

bool interpolate_method ( Object object, String method, Variant initial_val, Variant final_val, float times_in_sec, int trans_type, int ease_type, float delay=0 )

bool interpolate_property ( Object object, String property, Variant initial_val, Variant final_val, float times_in_sec, int trans_type, int ease_type, float delay=0 )

bool is_active ( ) const

bool is_repeat ( ) const

bool remove ( Object object, String key )

bool remove_all ( )

bool resume ( Object object, String key )

bool reset_all ( )

bool target_method ( Object object, String method, Object initial, String initial_method, Variant final_val, float times_in_sec, int trans_type, int ease_type, float delay=0 )

bool targeting_property ( Object object, String property, Object initial, String initial_val, Variant final_val, float times_in_sec, int trans_type, int ease_type, float delay=0 )

float tell ( ) const
```

12.346.3 Signals

- **tween_complete** ( Object object, String key )

  This signal is emitted when a tween ends.

- **tween_start** ( Object object, String key )

  This signal is emitted when a tween starts.

- **tween_step** ( Object object, String key, float elapsed, Object value )

  This signal is emitted each step of the tweening.
12.346.4 Numeric Constants

- **TWEEN_PROCESS_FIXED** = 0 — The Tween should use _fixed_process for timekeeping when this is enabled.
- **TWEEN_PROCESS_IDLE** = 1 — The Tween should use _process for timekeeping when this is enabled (default).
- **TRANS_LINEAR** = 0 — Means that the animation is interpolated linearly.
- **TRANS_SINE** = 1 — Means that the animation is interpolated using a sine wave.
- **TRANS_QUINT** = 2 — Means that the animation is interpolated with a quinary (to the power of 5) function.
- **TRANS_QUART** = 3 — Means that the animation is interpolated with a quartic (to the power of 4) function.
- **TRANS_QUAD** = 4 — Means that the animation is interpolated with a quadratic (to the power of 2) function.
- **TRANS_EXPO** = 5 — Means that the animation is interpolated with an exponential (some number to the power of x) function.
- **TRANS_ELASTIC** = 6 — Means that the animation is interpolated with elasticity, wiggling around the edges.
- **TRANS_CUBIC** = 7 — Means that the animation is interpolated with a cubic (to the power of 3) function.
- **TRANS_CIRC** = 8 — Means that the animation is interpolated with a function using square roots.
- **TRANS_BOUNCE** = 9 — Means that the animation is interpolated by bouncing at, but never surpassing, the end.
- **TRANS_BACK** = 10 — Means that the animation is interpolated backing out at edges.
- **EASE_IN** = 0 — Signifies that the interpolation should be focused in the beginning.
- **EASE_OUT** = 1 — Signifies that the interpolation should be focused in the end.
- **EASE_IN_OUT** = 2 — Signifies that the interpolation should be focused in both ends.
- **EASE_OUT_IN** = 3 — Signifies that the interpolation should be focused in both ends, but they should be switched (a bit hard to explain, try it for yourself to be sure).

12.346.5 Description

Node useful for animations with unknown start and end points, procedural animations, making one node follow another, and other simple behavior.

Because it is easy to get it wrong, here is a quick usage example:

```godot
define var tween = get_node("Tween")
tween.interpolate_property(get_node("Node2D_to_move"), "transform/pos", Vector2(0,0),
→   →Vector2(100,100), 1, Tween.TRANS_LINEAR, Tween.EASE_IN_OUT)
tween.start()
```

Some of the methods of this class require a property name. You can get the property name by hovering over the property in the inspector of the editor.

Many of the methods accept `trans_type` and `ease_type`. The first accepts an TRANS_* constant, and refers to the way the timing of the animation is handled (you might want to see http://easings.net/ for some examples). The second accepts an EASE_* constant, and controls where the `trans_type` is applied to the interpolation (in the beginning, the end, or both). If you don’t know which transision and easing to pick, you can try different TRANS_* constants with EASE_IN_OUT, and use the one that looks best.
12.346.6 Member Function Description

- **bool follow_method ( Object object, String method, Variant initial_val, Object target, String target_method, float times_in_sec, int trans_type, int ease_type, float delay=0 )**

Follow method of object and apply the returned value on target_method of target, beginning from initial_val for times_in_sec seconds, delay later. Methods are animated by calling them with consecutive values.

trans_type accepts TRANS_* constants, and is the way the animation is interpolated, while ease_type accepts EASE_* constants, and controls the place of the interpolation (the beginning, the end, or both). You can read more about them in the class description.

- **bool follow_property ( Object object, String property, Variant initial_val, Object target, String target_property, float times_in_sec, int trans_type, int ease_type, float delay=0 )**

Follow property of object and apply it on target_property of target, beginning from initial_val for times_in_sec seconds, delay seconds later. Note that target:target_property would equal object:property at the end of the tween.

trans_type accepts TRANS_* constants, and is the way the animation is interpolated, while ease_type accepts EASE_* constants, and controls the place of the interpolation (the beginning, the end, or both). You can read more about them in the class description.

- **float get_runtime ( ) const**

Returns the time needed for all tweens to end in seconds, measured from the start. Thus, if you have two tweens, one ending 10 seconds after the start and the other - 20 seconds, it would return 20 seconds, as by that time all tweens would have finished.

- **float get_speed ( ) const**

Returns the speed that has been set from editor GUI or set_repeat.

- **int get_tween_process_mode ( ) const**

Returns the process mode that has been set from editor GUI or set_tween_process_mode.

- **bool interpolate_callback ( Object object, float times_in_sec, String callback, Variant arg1=NULL, Variant arg2=NULL, Variant arg3=NULL, Variant arg4=NULL, Variant arg5=NULL )**

Call callback of object after times_in_sec. arg1-arg5 are arguments to be passed to the callback.

- **bool interpolate_deferred_callback ( Object object, float times_in_sec, String callback, Variant arg1=NULL, Variant arg2=NULL, Variant arg3=NULL, Variant arg4=NULL, Variant arg5=NULL )**

Call callback of object after times_in_sec on the main thread (similar to methog Object.call_deferred). [codearg1''-arg5 are arguments to be passed to the callback.

- **bool interpolate_method ( Object object, String method, Variant initial_val, Variant final_val, float times_in_sec, int trans_type, int ease_type, float delay=0 )**

Animate method of object from initial_val to final_val for times_in_sec seconds, delay seconds later. Methods are animated by calling them with consecutive values.

trans_type accepts TRANS_* constants, and is the way the animation is interpolated, while ease_type accepts EASE_* constants, and controls the place of the interpolation (the beginning, the end, or both). You can read more about them in the class description.

- **bool interpolate_property ( Object object, String property, Variant initial_val, Variant final_val, float times_in_sec, int trans_type, int ease_type, float delay=0 )**

Animate property of object from initial_val to final_val for times_in_sec seconds, delay seconds later.
trans_type accepts TRANS_* constants, and is the way the animation is interpolated, while ease_type accepts EASE_* constants, and controls the place of the interpolation (the begining, the end, or both). You can read more about them in the class description.

- **bool** is_active () const

Returns true if any tweens are currently running, and false otherwise. Note that this method doesn’t consider tweens that have ended.

- **bool** is_repeat () const

Returns true if repeat has been set from editor GUI or set_repeat.

- **bool** remove ( Object object, String key )

Stop animating and completely remove a tween, given its object and property/method pair.

- **bool** remove_all ()

Stop animating and completely remove all tweens.

- **bool** reset ( Object object, String key )

Resets a tween to the initial value (the one given, not the one before the tween), given its object and property/method pair.

- **bool** reset_all ()

Resets all tweens to their initial values (the ones given, not those before the tween).

- **bool** resume ( Object object, String key )

Continue animating a stopped tween, given its object and property/method pair.

- **bool** resume_all ()

Continue animating all stopped tweens.

- **bool** seek ( float time )

Seek the animation to the given time in seconds.

- **void** set_active ( bool active )

Activate/deactivate the tween. You can use this for pausing animations, though stop_all and resume_all might be more fit for this.

- **void** set_repeat ( bool repeat )

Make the tween repeat after all tweens have finished.

- **void** set_speed ( float speed )

Set the speed multiplier of the tween. Set it to 1 for normal speed, 2 for two times normal speed, and 0.5 for half of the normal speed. Setting it to 0 would pause the animation, but you might consider using set_active or stop_all and resume_all for this.

- **void** set_tween_process_mode ( int mode )

Set whether the Tween uses _process or _fixed_process (accepts TWEEN_PROCESS_IDLE and TWEEN_PROCESS_FIXED constants, respectively).

- **bool** start ()

Start the tween node. You can define tweens both before and after this.

- **bool** stop ( Object object, String key )

Stop animating a tween, given its object and property/method pair.
• `bool stop_all ()`
  Stop animating all tweens.

• `bool targeting_method ( Object object, String method, Object initial, String initial_method, Variant final_val, float times_in_sec, int trans_type, int ease_type, float delay=0 )`
  Animate method of object from the value returned by initial.initial_method to final_val for times_in_sec seconds, delay seconds later. Methods are animated by calling them with consecutive values.
  trans_type accepts TRANS_* constants, and is the way the animation is interpolated, while ease_type accepts EASE_* constants, and controls the place of the interpolation (the beginning, the end, or both). You can read more about them in the class description.

• `bool targeting_property ( Object object, String property, Object initial, String initial_val, Variant final_val, float times_in_sec, int trans_type, int ease_type, float delay=0 )`
  Animate property of object from the current value of the initial_val property of initial to final_val for times_in_sec seconds, delay seconds later.
  trans_type accepts TRANS_* constants, and is the way the animation is interpolated, while ease_type accepts EASE_* constants, and controls the place of the interpolation (the beginning, the end, or both). You can read more about them in the class description.

• `float tell () const`
  Returns the current time of the tween.

### 12.347 UndoRedo

Inherits: `Object`

Category: Core

#### 12.347.1 Brief Description

Helper to manage UndoRedo in the editor or custom tools.

#### 12.347.2 Member Functions

```
<table>
<thead>
<tr>
<th>Function</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void add_do_method ( Object object, String method, Variant arg0=NULL, Variant arg1=NULL, Variant arg2=NULL, Variant arg3=NULL, Variant arg4=NULL )</code></td>
<td></td>
</tr>
<tr>
<td><code>void add_do_property ( Object object, String property, Variant value )</code></td>
<td></td>
</tr>
<tr>
<td><code>void add_do_reference ( Object object )</code></td>
<td></td>
</tr>
<tr>
<td><code>void add_undo_method ( Object object, String method, Variant arg0=NULL, Variant arg1=NULL, Variant arg2=NULL, Variant arg3=NULL, Variant arg4=NULL )</code></td>
<td></td>
</tr>
<tr>
<td><code>void add_undo_property ( Object object, String property, Variant value )</code></td>
<td></td>
</tr>
<tr>
<td><code>void add_undo_reference ( Object object )</code></td>
<td></td>
</tr>
<tr>
<td><code>void clear_history ()</code></td>
<td></td>
</tr>
<tr>
<td><code>void commit_action ()</code></td>
<td></td>
</tr>
<tr>
<td><code>void create_action ( String name, int merge_mode=0 )</code></td>
<td></td>
</tr>
<tr>
<td><code>String get_current_action_name () const</code></td>
<td></td>
</tr>
<tr>
<td><code>int get_version () const</code></td>
<td></td>
</tr>
</tbody>
</table>
```
12.347.3 Numeric Constants

- MERGE_DISABLE = 0
- MERGE_ENDS = 1
- MERGE_ALL = 2

12.347.4 Description

Helper to manage UndoRedo in the editor or custom tools. It works by storing calls to functions in both ‘do’ an ‘undo’ lists.

Common behavior is to create an action, then add do/undo calls to functions or property changes, then committing the action.

12.347.5 Member Function Description

- void add_do_method ( Object object, String method, Variant arg0=NULL, Variant arg1=NULL, Variant arg2=NULL, Variant arg3=NULL, Variant arg4=NULL )
  Add a call to a method in a given object with custom arguments.
  - void add_do_property ( Object object, String property, Variant value )
    Set a property with a custom value.
  - void add_do_reference ( Object object )
    Add a ‘do’ reference that will be erased if the ‘do’ history is lost. This is useful mostly for new nodes created for the ‘do’ call. Do not use for resources.
  - void add_undo_method ( Object object, String method, Variant arg0=NULL, Variant arg1=NULL, Variant arg2=NULL, Variant arg3=NULL, Variant arg4=NULL )
    Add a call to an undo method in a given object with custom arguments. Undo calls are used to revert ‘do’ calls.
  - void add_undo_property ( Object object, String property, Variant value )
    Undo setting of a property with a custom value.
  - void add_undo_reference ( Object object )
    Add an ‘undo’ reference that will be erased if the ‘undo’ history is lost. This is useful mostly for nodes removed with the ‘do’ call (not the ‘undo’ call!).
  - void clear_history ( )
    Clear the undo/redo history and associated references.
  - void commit_action ( )
    Commit the action. All ‘do’ methods/properties are called/set when this function is called.
  - void create_action ( String name, int merge_mode=0 )
    Create a new action. After this is called, do all your calls to add_do_method, add_undo_method, add_do_property and add_undo_property.
  - String get_current_action_name ( ) const
Get the name of the current action.

- \texttt{int get\_version()} \texttt{const}

Get the version, each time a new action is committed, the version number of the UndoRedo is increased automatically. This is useful mostly to check if something changed from a saved version.

\section*{12.348 Variant}

\textbf{Category:} Core

\subsection*{12.348.1 Brief Description}

\section*{12.349 VBoxContainer}

\textbf{Inherits:} BoxContainer $<$ Container $<$ Control $<$ CanvasItem $<$ Node $<$ Object

\textbf{Category:} Core

\subsection*{12.349.1 Brief Description}

Vertical box container.

\subsection*{12.349.2 Description}

Vertical box container. See BoxContainer.

\section*{12.350 VButtonArray}

\textbf{Inherits:} ButtonArray $<$ Control $<$ CanvasItem $<$ Node $<$ Object

\textbf{Category:} Core

\subsection*{12.350.1 Brief Description}

Vertical button array.

\subsection*{12.350.2 Description}

Vertical button array. See ButtonArray.

\section*{12.351 Vector2}

\textbf{Category:} Built-In Types
12.351.1 Brief Description

Vector used for 2D Math.

12.351.2 Member Functions

<table>
<thead>
<tr>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vector2 (float x, float y)</td>
</tr>
<tr>
<td>abs ()</td>
</tr>
<tr>
<td>angle ()</td>
</tr>
<tr>
<td>angle_to (Vector2 to)</td>
</tr>
<tr>
<td>angle_to_point (Vector2 to)</td>
</tr>
<tr>
<td>clamped (float length)</td>
</tr>
<tr>
<td>cubic_interpolate (Vector2 b, Vector2 pre_a, Vector2 post_b, float t)</td>
</tr>
<tr>
<td>distance_squared_to (Vector2 to)</td>
</tr>
<tr>
<td>distance_to (Vector2 to)</td>
</tr>
<tr>
<td>dot (Vector2 with)</td>
</tr>
<tr>
<td>floor ()</td>
</tr>
<tr>
<td>floorf ()</td>
</tr>
<tr>
<td>get_aspect ()</td>
</tr>
<tr>
<td>length ()</td>
</tr>
<tr>
<td>length_squared ()</td>
</tr>
<tr>
<td>linear_interpolate (Vector2 b, float t)</td>
</tr>
<tr>
<td>normalized ()</td>
</tr>
<tr>
<td>reflect (Vector2 vec)</td>
</tr>
<tr>
<td>rotated (float phi)</td>
</tr>
<tr>
<td>slide (Vector2 vec)</td>
</tr>
<tr>
<td>snapped (Vector2 by)</td>
</tr>
<tr>
<td>tangent ()</td>
</tr>
</tbody>
</table>

12.351.3 Member Variables

- float height - Height of the vector (Same as Y).
- float width - Width of the vector (Same as X).
- float x - X component of the vector.
- float y - Y component of the vector.

12.351.4 Description

2-element structure that can be used to represent positions in 2d-space, or any other pair of numeric values.

12.351.5 Member Function Description

- Vector2 Vector2 (float x, float y)
  Constructs a new Vector2 from the given x and y.
- Vector2 abs ()
  Returns a new vector with all components in absolute values (i.e. positive).
• float angle ( )
Returns the result of atan2 when called with the Vector’s x and y as parameters (Math::atan2(x,y)).
Be aware that it therefore returns an angle oriented clockwise with regard to the (0, 1) unit vector, and not an angle oriented counter-clockwise with regard to the (1, 0) unit vector (which would be the typical trigonometric representation of the angle when calling Math::atan2(y,x)).

• float angle_to ( Vector2 to )
Returns the angle in radians between the two vectors.

• float angle_to_point ( Vector2 to )
Returns the angle in radians between the line connecting the two points and the x coordinate.

• Vector2 clamped ( float length )
Returns the vector with a maximum length.

• Vector2 cubic_interpolate ( Vector2 b, Vector2 pre_a, Vector2 post_b, float t )
Cubicly interpolates between this Vector and “b”, using “pre_a” and “post_b” as handles, and returning the result at position “t”.

• float distance_squared_to ( Vector2 to )
Returns the squared distance to vector “b”. Prefer this function over “distance_to” if you need to sort vectors or need the squared distance for some formula.

• float distance_to ( Vector2 to )
Returns the distance to vector “b”.

• float dot ( Vector2 with )
Returns the dot product with vector “b”.

• Vector2 floor ( )
Remove the fractional part of x and y.

• Vector2 floorf ( )
Remove the fractional part of x and y.

• float get_aspect ( )
Returns the ratio of X to Y.

• float length ( )
Returns the length of the vector.

• float length_squared ( )
Returns the squared length of the vector. Prefer this function over “length” if you need to sort vectors or need the squared length for some formula.

• Vector2 linear_interpolate ( Vector2 b, float t )
Returns the result of the linear interpolation between this vector and “b”, by amount “t”.

• Vector2 normalized ( )
Returns a normalized vector to unit length.

• Vector2 reflect ( Vector2 vec )
Like “slide”, but reflects the Vector instead of continuing along the wall.
• **Vector2 rotated ( float phi )**
  Rotates the vector by “phi” radians.
  
• **Vector2 slide ( Vector2 vec )**
  Slides the vector by the other vector.
  
• **Vector2 snapped ( Vector2 by )**
  Snaps the vector to a grid with the given size.
  
• **Vector2 tangent ( )**
  Returns a perpendicular vector.

## 12.352 Vector2Array

**Category:** Built-In Types

### 12.352.1 Brief Description

An Array of Vector2.

### 12.352.2 Member Functions

<table>
<thead>
<tr>
<th>Vector2Array</th>
<th>Vector2Array ( Array from )</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>append ( Vector2 vector2 )</td>
</tr>
<tr>
<td>void</td>
<td>append_array ( Vector2Array array )</td>
</tr>
<tr>
<td>int</td>
<td>insert ( int idx, Vector2 vector2 )</td>
</tr>
<tr>
<td>void</td>
<td>invert ( )</td>
</tr>
<tr>
<td>void</td>
<td>push_back ( Vector2 vector2 )</td>
</tr>
<tr>
<td>void</td>
<td>remove ( int idx )</td>
</tr>
<tr>
<td>void</td>
<td>resize ( int idx )</td>
</tr>
<tr>
<td>void</td>
<td>set ( int idx, Vector2 vector2 )</td>
</tr>
<tr>
<td>int</td>
<td>size ( )</td>
</tr>
</tbody>
</table>

### 12.352.3 Description

An Array specifically designed to hold Vector2.

### 12.352.4 Member Function Description

• **Vector2Array Vector2Array ( Array from )**
  Construct a new Vector2Array. Optionally, you can pass in an Array that will be converted.
  
• void **append ( Vector2 vector2 )**
  Append an element at the end of the array (alias of push_back).
  
• void **append_array ( Vector2Array array )**
  Append an Vector2Array at the end of this array.
• `int insert(int idx, Vector2 vector2)`

Insert a new element at a given position in the array. The position must be valid, or at the end of the array (pos==size()).

• `void invert()`

Reverse the order of the elements in the array (so first element will now be the last).

• `void push_back(Vector2 vector2)`

Insert a `Vector2` at the end.

• `void remove(int idx)`

Remove an element from the array by index.

• `void resize(int idx)`

Set the size of the Vector2Array. If larger than the current size it will reserve some space beforehand, and if it is smaller it will cut off the array.

• `void set(int idx, Vector2 vector2)`

Change the `Vector2` at the given index.

• `int size()`

Return the size of the array.

## 12.353 Vector3

**Category:** Built-In Types

### 12.353.1 Brief Description

Vector class, which performs basic 3D vector math operations.
12.353.2 Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vector3 ( float x, float y, float z )</td>
<td>Returns a Vector3 with the given components.</td>
</tr>
<tr>
<td>Vector3 abs ( )</td>
<td></td>
</tr>
<tr>
<td>float angle_to ( Vector3 to )</td>
<td></td>
</tr>
<tr>
<td>Vector3 ceil ( )</td>
<td></td>
</tr>
<tr>
<td>Vector3 cross ( Vector3 b )</td>
<td></td>
</tr>
<tr>
<td>Vector3 cubic_interpolate ( Vector3 b, Vector3 pre_a, Vector3 post_b, float t )</td>
<td></td>
</tr>
<tr>
<td>float distance_squared_to ( Vector3 b )</td>
<td></td>
</tr>
<tr>
<td>float distance_to ( Vector3 b )</td>
<td></td>
</tr>
<tr>
<td>float dot ( Vector3 b )</td>
<td></td>
</tr>
<tr>
<td>Vector3 floor ( )</td>
<td></td>
</tr>
<tr>
<td>Vector3 inverse ( )</td>
<td></td>
</tr>
<tr>
<td>float length ( )</td>
<td></td>
</tr>
<tr>
<td>float length_squared ( )</td>
<td></td>
</tr>
<tr>
<td>Vector3 linear_interpolate ( Vector3 b, float t )</td>
<td></td>
</tr>
<tr>
<td>int max_axis ( )</td>
<td></td>
</tr>
<tr>
<td>int min_axis ( )</td>
<td></td>
</tr>
<tr>
<td>Vector3 normalized ( )</td>
<td></td>
</tr>
<tr>
<td>Vector3 reflect ( Vector3 by )</td>
<td></td>
</tr>
<tr>
<td>Vector3 rotated ( Vector3 axis, float phi )</td>
<td></td>
</tr>
<tr>
<td>Vector3 slide ( Vector3 by )</td>
<td></td>
</tr>
<tr>
<td>Vector3 snapped ( float by )</td>
<td></td>
</tr>
</tbody>
</table>

12.353.3 Member Variables

- **float x** - X component of the vector.
- **float y** - Y component of the vector.
- **float z** - Z component of the vector.

12.353.4 Numeric Constants

- **AXIS_X = 0** — Enumerated value for the X axis. Returned by functions like max_axis or min_axis.
- **AXIS_Y = 1** — Enumerated value for the Y axis.
- **AXIS_Z = 2** — Enumerated value for the Z axis.

12.353.5 Description

Vector3 is one of the core classes of the engine, and includes several built-in helper functions to perform basic vector math operations.

12.353.6 Member Function Description

- **Vector3 Vector3 ( float x, float y, float z )**

Returns a Vector3 with the given components.

- **Vector3 abs ( )**
Returns a new vector with all components in absolute values (i.e. positive).

- `float angle_to ( Vector3 to )`
- `Vector3 ceil ( )`

Returns a new vector with all components rounded up.

- `Vector3 cross ( Vector3 b )`

Return the cross product with b.

- `Vector3 cubic_interpolate ( Vector3 b, Vector3 pre_a, Vector3 post_b, float t )`

Perform a cubic interpolation between vectors pre_a, a, b, post_b (a is current), by the given amount (t).

- `float distance_squared_to ( Vector3 b )`

Return the squared distance (distance minus the last square root) to b. Prefer this function over distance_to if you need to sort vectors or need the squared distance for some formula.

- `float distance_to ( Vector3 b )`

Return the distance to b.

- `float dot ( Vector3 b )`

Return the dot product with b.

- `Vector3 floor ( )`

Returns a new vector with all components rounded down.

- `Vector3 inverse ( )`

Returns the inverse of the vector. This is the same as Vector3( 1.0 / v.x, 1.0 / v.y, 1.0 / v.z )

- `float length ( )`

Return the length of the vector.

- `float length_squared ( )`

Return the length of the vector, squared. Prefer this function over “length” if you need to sort vectors or need the squared length for some formula.

- `Vector3 linear_interpolate ( Vector3 b, float t )`

Linearly interpolates the vector to a given one (b), by the given amount (t).

- `int max_axis ( )`

Returns AXIS_X, AXIS_Y or AXIS_Z depending on which axis is the largest.

- `int min_axis ( )`

Returns AXIS_X, AXIS_Y or AXIS_Z depending on which axis is the smallest.

- `Vector3 normalized ( )`

Return a copy of the normalized vector to unit length. This is the same as v / v.length().

- `Vector3 reflect ( Vector3 by )`

Like “slide”, but reflects the Vector instead of continuing along the wall.

- `Vector3 rotated ( Vector3 axis, float phi )`

Rotates the vector around some axis by phi radians.

- `Vector3 slide ( Vector3 by )`
Slides the vector along a wall.

- **Vector3 snapped** (*float by*)

  Return a copy of the vector, snapped to the lowest neared multiple.

### 12.354 Vector3Array

**Category:** Built-In Types

#### 12.354.1 Brief Description

An Array of Vector3.

#### 12.354.2 Member Functions

<table>
<thead>
<tr>
<th>Vector3Array</th>
<th>Vector3Array (Array from)</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>append (Vector3 vector3)</td>
</tr>
<tr>
<td>void</td>
<td>append_array (Vector3Array array)</td>
</tr>
<tr>
<td>int</td>
<td>insert (int idx, Vector3 vector3)</td>
</tr>
<tr>
<td>void</td>
<td>invert ()</td>
</tr>
<tr>
<td>void</td>
<td>push_back (Vector3 vector3)</td>
</tr>
<tr>
<td>void</td>
<td>remove (int idx)</td>
</tr>
<tr>
<td>void</td>
<td>resize (int idx)</td>
</tr>
<tr>
<td>void</td>
<td>set (int idx, Vector3 vector3)</td>
</tr>
<tr>
<td>int</td>
<td>size ()</td>
</tr>
</tbody>
</table>

#### 12.354.3 Description

An Array specifically designed to hold Vector3.

#### 12.354.4 Member Function Description

- **Vector3Array Vector3Array (Array from)**

  Construct a new Vector3Array. Optionally, you can pass in an Array that will be converted.

- **void append (Vector3 vector3)**

  Append an element at the end of the array (alias of push_back).

- **void append_array (Vector3Array array)**

  Append an Vector3Array at the end of this array.

- **int insert (int idx, Vector3 vector3)**

  Insert a new element at a given position in the array. The position must be valid, or at the end of the array (pos==size()).

- **void invert ()**

  Reverse the order of the elements in the array (so first element will now be the last).
• void **push_back** ( Vector3 vector3 )

Insert a Vector3 at the end.

• void **remove** ( int idx )

Remove an element from the array by index.

• void **resize** ( int idx )

Set the size of the Vector3Array. If larger than the current size it will reserve some space beforehand, and if it is smaller it will cut off the array.

• void **set** ( int idx, Vector3 vector3 )

Change the Vector3 at the given index.

• int **size** ( )

Return the size of the array.

### 12.355 VehicleBody

**Inherits:** PhysicsBody < CollisionObject < Spatial < Node < Object

**Category:** Core

#### 12.355.1 Brief Description

#### 12.355.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>float</td>
<td>get_brake ( ) const</td>
<td></td>
</tr>
<tr>
<td>float</td>
<td>get_engine_force ( ) const</td>
<td></td>
</tr>
<tr>
<td>float</td>
<td>get_friction ( ) const</td>
<td></td>
</tr>
<tr>
<td>Vector3</td>
<td>get_linear_velocity ( ) const</td>
<td></td>
</tr>
<tr>
<td>float</td>
<td>get_mass ( ) const</td>
<td></td>
</tr>
<tr>
<td>float</td>
<td>get_steering ( ) const</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>set_brake ( float brake )</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>set_engine_force ( float engine_force )</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>set_friction ( float friction )</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>set_mass ( float mass )</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>set_steering ( float steering )</td>
<td></td>
</tr>
</tbody>
</table>

#### 12.355.3 Member Function Description

• **float get_brake ( ) const**

• **float get_engine_force ( ) const**

• **float get_friction ( ) const**

• **Vector3 get_linear_velocity ( ) const**

Returns the VehicleBody’s velocity vector. To get the absolute speed in scalar value, get the length of the return vector in pixels/second. Example:
Godot Engine Documentation, Release 2.1

```godot
# vehicle is an instance of VehicleBody
var speed = vehicle.get_linear_velocity().length()
```

- `float get_mass ( ) const`  
- `float get_steering ( ) const`  
- `void set_brake ( float brake )`  
- `void set_engine_force ( float engine_force )`  
- `void set_friction ( float friction )`  
- `void set_mass ( float mass )`  
- `void set_steering ( float steering )`

### 12.356 VehicleWheel

*Inherits: Spatial < Node < Object*

**Category:** Core

#### 12.356.1 Brief Description

#### 12.356.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>float</td>
<td>get_damping_compression ( ) const</td>
</tr>
<tr>
<td>float</td>
<td>get_damping_relaxation ( ) const</td>
</tr>
<tr>
<td>float</td>
<td>get_friction_slip ( ) const</td>
</tr>
<tr>
<td>float</td>
<td>get_radius ( ) const</td>
</tr>
<tr>
<td>float</td>
<td>get_roll_influence ( ) const</td>
</tr>
<tr>
<td>float</td>
<td>get_suspension_max_force ( ) const</td>
</tr>
<tr>
<td>float</td>
<td>get_suspension_rest_length ( ) const</td>
</tr>
<tr>
<td>float</td>
<td>get_suspension_stiffness ( ) const</td>
</tr>
<tr>
<td>float</td>
<td>get_suspension_travel ( ) const</td>
</tr>
<tr>
<td>bool</td>
<td>is_in_contact ( ) const</td>
</tr>
<tr>
<td>bool</td>
<td>is_used_as_steering ( ) const</td>
</tr>
<tr>
<td>bool</td>
<td>is_used_as_traction ( ) const</td>
</tr>
<tr>
<td>void</td>
<td>set_damping_compression ( float length )</td>
</tr>
<tr>
<td>void</td>
<td>set_damping_relaxation ( float length )</td>
</tr>
<tr>
<td>void</td>
<td>set_friction_slip ( float length )</td>
</tr>
<tr>
<td>void</td>
<td>set_radius ( float length )</td>
</tr>
<tr>
<td>void</td>
<td>set_roll_influence ( float roll_influence )</td>
</tr>
<tr>
<td>void</td>
<td>set_suspension_max_force ( float length )</td>
</tr>
<tr>
<td>void</td>
<td>set_suspension_rest_length ( float length )</td>
</tr>
<tr>
<td>void</td>
<td>set_suspension_stiffness ( float length )</td>
</tr>
<tr>
<td>void</td>
<td>set_suspension_travel ( float length )</td>
</tr>
<tr>
<td>void</td>
<td>set_use_as_steering ( bool enable )</td>
</tr>
<tr>
<td>void</td>
<td>set_use_as_traction ( bool enable )</td>
</tr>
</tbody>
</table>
12.356.3 Member Function Description

- float get_damping_compression ( ) const
- float get_damping_relaxation ( ) const
- float get_friction_slip ( ) const
- float get_radius ( ) const
- float get_roll_influence ( ) const
- float getSuspensionMaxForce ( ) const
- float getSuspensionRestLength ( ) const
- float getSuspensionStiffness ( ) const
- float getSuspensionTravel ( ) const
- bool is_in_contact ( ) const
- bool is_used_as_steering ( ) const
- bool is_used_as_traction ( ) const
- void set_damping_compression ( float length )
- void set_damping_relaxation ( float length )
- void set_friction_slip ( float length )
- void set_radius ( float length )
- void set_roll_influence ( float roll_influence )
- void setSuspensionMaxForce ( float length )
- void setSuspensionRestLength ( float length )
- void setSuspensionStiffness ( float length )
- void setSuspensionTravel ( float length )
- void set_use_as_steering ( bool enable )
- void set_use_as_traction ( bool enable )

12.357 VideoPlayer

Inherits: Control < CanvasItem < Node < Object

Category: Core

12.357.1 Brief Description

Control to play video files.
12.357.2 Member Functions

<table>
<thead>
<tr>
<th>data type</th>
<th>function name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>get_audio_track()</td>
<td>Get the selected audio track (for multitrack videos).</td>
</tr>
<tr>
<td>int</td>
<td>get_buffering_msec()</td>
<td>Get the amount of milliseconds to store in buffer while playing.</td>
</tr>
<tr>
<td>VideoStream</td>
<td>get_stream()</td>
<td>Get the video stream.</td>
</tr>
<tr>
<td>String</td>
<td>get_stream_name()</td>
<td>Get the name of the video stream.</td>
</tr>
<tr>
<td>float</td>
<td>get_stream_pos()</td>
<td>Get the current position of the stream, in seconds.</td>
</tr>
<tr>
<td>Texture</td>
<td>get_video_texture()</td>
<td>Get the current frame of the video as a Texture.</td>
</tr>
<tr>
<td>float</td>
<td>get_volume()</td>
<td>Get the volume ( ) const</td>
</tr>
<tr>
<td>float</td>
<td>get_volume_db()</td>
<td>Get the volume in dB ( ) const</td>
</tr>
<tr>
<td>bool</td>
<td>has_autoplay()</td>
<td>Get the autoplay ( ) const</td>
</tr>
<tr>
<td>bool</td>
<td>has_expanded()</td>
<td>Get the expanded ( ) const</td>
</tr>
<tr>
<td>bool</td>
<td>is_paused()</td>
<td>Get the paused ( ) const</td>
</tr>
<tr>
<td>bool</td>
<td>is_playing()</td>
<td>Get the playing ( ) const</td>
</tr>
<tr>
<td>void</td>
<td>play()</td>
<td>Play the video.</td>
</tr>
<tr>
<td>void</td>
<td>set_audio_track(track)</td>
<td>Set the audio track.</td>
</tr>
<tr>
<td>void</td>
<td>set_autoplay(enabled)</td>
<td>Set the autoplay.</td>
</tr>
<tr>
<td>void</td>
<td>set_buffering_msec(msec)</td>
<td>Set the buffering msec.</td>
</tr>
<tr>
<td>void</td>
<td>set_expanded(enable)</td>
<td>Set the expanded.</td>
</tr>
<tr>
<td>void</td>
<td>set_paused(paused)</td>
<td>Set the paused.</td>
</tr>
<tr>
<td>void</td>
<td>set_stream(stream)</td>
<td>Set the stream.</td>
</tr>
<tr>
<td>void</td>
<td>set_volume(volume)</td>
<td>Set the volume.</td>
</tr>
<tr>
<td>void</td>
<td>set_volume_db(db)</td>
<td>Set the volume in dB.</td>
</tr>
<tr>
<td>void</td>
<td>stop()</td>
<td>Stop the video.</td>
</tr>
</tbody>
</table>

12.357.3 Description

This control has the ability to play video streams. The only format accepted is the OGV Theora, so any other format must be converted before using in a project.

12.357.4 Member Function Description

- **int get_audio_track() const**
  Get the selected audio track (for multitrack videos).

- **int get_buffering_msec() const**
  Get the amount of milliseconds to store in buffer while playing.

- **VideoStream get_stream() const**
  Get the video stream.

- **String get_stream_name() const**
  Get the name of the video stream.

- **float get_stream_pos() const**
  Get the current position of the stream, in seconds.

- **Texture get_video_texture()**
  Get the current frame of the video as a Texture.

- **float get_volume() const**
Get the volume of the audio track as a linear value.

- float get_volume_db() const

Get the volume of the audio track in decibels.

- bool has_autoplay() const

Get whether or not the video is set as autoplay.

- bool has_expand() const

Get whether or not the expand property is set.

- bool is_paused() const

Get whether or not the video is paused.

- bool is_playing() const

Get whether or not the video is playing.

- void play()

Start the video playback.

- void set_audio_track(int track)

Set the audio track (for multitrack videos).

- void set_autoplay(bool enabled)

Set whether this node should start playing automatically.

- void set_buffering_msec(int msec)

Set the amount of milliseconds to buffer during playback.

- void set_expand(bool enable)

Set the expand property. If enabled, the video will grow or shrink to fit the player size, otherwise it will play at the stream resolution.

- void set_paused(bool paused)

Set whether the video should pause the playback.

- void set_stream(VideoStream stream)

Set the video stream for this player.

- void set_volume(float volume)

Set the audio volume as a linear value.

- void set_volume_db(float db)

Set the audio volume in decibels.

- void stop()

Stop the video playback.

### 12.358 VideoStream

**Inherits:** Resource < Reference < Object

**Inherited By:** VideoStreamTheora
Godot Engine Documentation, Release 2.1

Category: Core

12.358.1 Brief Description

12.359 VideoStreamTheora

Inherits: VideoStream < Resource < Reference < Object

Category: Core

12.359.1 Brief Description

12.360 Viewport

Inherits: Node < Object

Category: Core

12.360.1 Brief Description

Creates a sub-view into the screen.

12.360.2 Member Functions

<table>
<thead>
<tr>
<th>World</th>
<th>find_world ( ) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>World2D</td>
<td>find_world_2d ( ) const</td>
</tr>
<tr>
<td>Camera</td>
<td>get_camera ( ) const</td>
</tr>
<tr>
<td>Matrix32</td>
<td>get_canvas_transform ( ) const</td>
</tr>
<tr>
<td>Matrix32</td>
<td>get_final_transform ( ) const</td>
</tr>
<tr>
<td>Matrix32</td>
<td>get_global_canvas_transform ( ) const</td>
</tr>
<tr>
<td>Vector2</td>
<td>get_mouse_pos ( ) const</td>
</tr>
<tr>
<td>bool</td>
<td>get_physics_object_picking ( )</td>
</tr>
<tr>
<td>Rect2</td>
<td>get_rect ( ) const</td>
</tr>
<tr>
<td>bool</td>
<td>get_render_target_clear_on_new_frame ( ) const</td>
</tr>
<tr>
<td>bool</td>
<td>get_render_target_filter ( ) const</td>
</tr>
<tr>
<td>bool</td>
<td>get_render_target_gen_mipmaps ( ) const</td>
</tr>
<tr>
<td>RenderTargetTexture</td>
<td>get_render_target_texture ( ) const</td>
</tr>
<tr>
<td>int</td>
<td>get_render_target_update_mode ( ) const</td>
</tr>
<tr>
<td>bool</td>
<td>get_render_target_vflip ( ) const</td>
</tr>
<tr>
<td>Image</td>
<td>get_screen_capture ( ) const</td>
</tr>
<tr>
<td>Vector2</td>
<td>get_size_override ( ) const</td>
</tr>
<tr>
<td>RID</td>
<td>get_viewport ( ) const</td>
</tr>
<tr>
<td>Rect2</td>
<td>get_visible_rect ( ) const</td>
</tr>
<tr>
<td>World</td>
<td>get_world ( ) const</td>
</tr>
<tr>
<td>World2D</td>
<td>get_world_2d ( ) const</td>
</tr>
<tr>
<td>Variant</td>
<td>gui_get_drag_data ( ) const</td>
</tr>
<tr>
<td>bool</td>
<td>gui_has_modal_stack ( ) const</td>
</tr>
</tbody>
</table>

Continued on next page
### Table 12.41 – continued from previous page

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>has_transparent_background( ) const</td>
<td></td>
</tr>
<tr>
<td>input( InputEvent local_event )</td>
<td></td>
</tr>
<tr>
<td>is_audio_listener( ) const</td>
<td></td>
</tr>
<tr>
<td>is_audio_listener_2d( ) const</td>
<td></td>
</tr>
<tr>
<td>is_input_disabled( ) const</td>
<td></td>
</tr>
<tr>
<td>is_set_as_render_target( ) const</td>
<td></td>
</tr>
<tr>
<td>is_size_override_enabled( ) const</td>
<td></td>
</tr>
<tr>
<td>is_size_override_stretch_enabled( ) const</td>
<td></td>
</tr>
<tr>
<td>is_using_own_world( ) const</td>
<td></td>
</tr>
<tr>
<td>queue_screen_capture( )</td>
<td></td>
</tr>
<tr>
<td>render_target_clear( )</td>
<td></td>
</tr>
<tr>
<td>set_as_audio_listener( bool enable )</td>
<td></td>
</tr>
<tr>
<td>set_as_audio_listener_2d( bool enable )</td>
<td></td>
</tr>
<tr>
<td>set_as_render_target( bool enable )</td>
<td></td>
</tr>
<tr>
<td>set_canvas_transform( Matrix32 xform )</td>
<td></td>
</tr>
<tr>
<td>set_disable_input( bool disable )</td>
<td></td>
</tr>
<tr>
<td>set_global_canvas_transform( Matrix32 xform )</td>
<td></td>
</tr>
<tr>
<td>set_physics_object_picking( bool enable )</td>
<td></td>
</tr>
<tr>
<td>set_rect( Rect2 rect )</td>
<td></td>
</tr>
<tr>
<td>set_render_target_clear_on_new_frame( bool enable )</td>
<td></td>
</tr>
<tr>
<td>set_render_target_filter( bool enable )</td>
<td></td>
</tr>
<tr>
<td>set_render_target_gen_mipmaps( bool enable )</td>
<td></td>
</tr>
<tr>
<td>set_render_target_to_screen_rect( Rect2 rect )</td>
<td></td>
</tr>
<tr>
<td>set_render_target_update_mode( int mode )</td>
<td></td>
</tr>
<tr>
<td>set_render_target_vflip( bool enable )</td>
<td></td>
</tr>
<tr>
<td>set_size_override( bool enable, Vector2 size=Vector2(-1, -1), Vector2 margin=Vector2(0, 0) )</td>
<td></td>
</tr>
<tr>
<td>set_size_override_stretch( bool enabled )</td>
<td></td>
</tr>
<tr>
<td>set_transparent_background( bool enable )</td>
<td></td>
</tr>
<tr>
<td>set_use_own_world( bool enable )</td>
<td></td>
</tr>
<tr>
<td>set_world( World world )</td>
<td></td>
</tr>
<tr>
<td>set_world_2d( World2D world_2d )</td>
<td></td>
</tr>
<tr>
<td>unhandled_input( InputEvent local_event )</td>
<td></td>
</tr>
<tr>
<td>update_worlds( )</td>
<td></td>
</tr>
<tr>
<td>warp_mouse( Vector2 to_pos )</td>
<td></td>
</tr>
</tbody>
</table>

#### 12.360.3 Signals

- **size_changed( )**

  Emitted when the size of the viewport is changed, whether by `set_size_override`, resize of window, or some other means.

#### 12.360.4 Numeric Constants

- **RENDER_TARGET_UPDATE_DISABLED = 0** — Do not update the render target.
- **RENDER_TARGET_UPDATE_ONCE = 1** — Update the render target once, then switch to `RENDER_TARGET_UPDATE_DISABLED`
- **RENDER_TARGET_UPDATE_WHEN_VISIBLE = 2** — Update the render target only when it is visible.
  
  This is the default value.
• RENDER_TARGET_UPDATE_ALWAYS = 3 — Update the render target always.

12.360.5 Description

A Viewport creates a different view into the screen, or a sub-view inside another viewport. Children 2D Nodes will display on it, and children Camera 3D nodes will render on it too.

Optionally, a viewport can have its own 2D or 3D world, so they don’t share what they draw with other viewports.

If a viewport is a child of a Control, it will automatically take up its same rect and position, otherwise they must be set manually.

Viewports can also choose to be audio listeners, so they generate positional audio depending on a 2D or 3D camera child of it.

Also, viewports can be assigned to different screens in case the devices have multiple screens.

Finally, viewports can also behave as render targets, in which case they will not be visible unless the associated texture is used to draw.

12.360.6 Member Function Description

• World find_world ( ) const
  Return the 3D world of the viewport, or if no such present, the one of the parent viewport.

• World2D find_world_2d ( ) const
  Return the 2D world of the viewport.

• Camera get_camera ( ) const
  Return the active 3D camera.

• Matrix32 get_canvas_transform ( ) const
  Get the canvas transform of the viewport.

• Matrix32 get_final_transform ( ) const
  Get the total transform of the viewport.

• Matrix32 get_global_canvas_transform ( ) const
  Get the global canvas transform of the viewport.

• Vector2 get_mouse_pos ( ) const
  Get the mouse position, relative to the viewport.

• bool get_physics_object_picking ( )
  Get whether picking for all physics objects inside the viewport is enabled.

• Rect2 get_rect ( ) const
  Return the viewport rect. If the viewport is child of a control, it will use the same rect as the parent. Otherwise, if the rect is empty, the viewport will use all the allowed space.

• bool get_render_target_clear_on_new_frame ( ) const
  Return whether automatic clearing of the render target on each frame is enabled.

• bool get_render_target_filter ( ) const
Get whether the rendered texture has filters enabled.

- \texttt{bool get\_render\_target\_gen\_mipmaps()} \texttt{const}

Get whether the rendered texture will have mipmaps generated.

- \texttt{RenderTargetTexture get\_render\_target\_texture()} \texttt{const}

Get the render target’s texture, for use with various objects that you want to texture with the viewport.

- \texttt{int get\_render\_target\_update\_mode()} \texttt{const}

Get when the render target would be updated, will be one of the \texttt{RENDER\_TARGET\_UPDATE\_\_\_\_} constants.

- \texttt{bool get\_render\_target\_vflip()} \texttt{const}

Set whether the render target is flipped on the Y axis.

- \texttt{Image get\_screen\_capture()} \texttt{const}

Return the captured screenshot after \texttt{queue_screen\_capture}. You might need to check more than one frame untill the right image is returned.

- \texttt{Vector2 get\_size\_override()} \texttt{const}

Get the size override set with \texttt{set\_size\_override}.

- \texttt{RID get\_viewport()} \texttt{const}

Get the viewport RID from the visual server.

- \texttt{Rect2 get\_visible\_rect()} \texttt{const}

Return the final, visible rect in global screen coordinates.

- \texttt{World get\_world()} \texttt{const}

Return the 3D world of the viewport.

- \texttt{World2D get\_world\_2d()} \texttt{const}

- \texttt{Variant gui\_get\_drag\_data()} \texttt{const}

Returns the drag data from the GUI, that was previously returned by \texttt{Control\_get\_drag\_data}.

- \texttt{bool gui\_has\_modal\_stack()} \texttt{const}

Returns whether there are shown modals on-screen.

- \texttt{bool has\_transparent\_background()} \texttt{const}

Return whether the viewport lets whatever is behind it to show.

- \texttt{void input(InputEvent local\_event)}

- \texttt{bool is\_audio\_listener()} \texttt{const}

Returns whether the viewport sends sounds to the speakers.

- \texttt{bool is\_audio\_listener\_2d()} \texttt{const}

Returns whether the viewport sends sounds from 2D emitters to the speakers.

- \texttt{bool is\_input\_disabled()} \texttt{const}

Return whether input to the viewport is disabled.

- \texttt{bool is\_set\_as\_render\_target()} \texttt{const}

Return whether the viewport is set as a render target by \texttt{set\_as\_render\_target}. 
• **bool** is_size_override_enabled ( ) const

Get the enabled status of the size override set with **set_size_override**.

• **bool** is_size_override_stretch_enabled ( ) const

Get the enabled status of the size stretch override set with **set_size_override_stretch**.

• **bool** is_using_own_world ( ) const

Return whether the viewport is using a world separate from the parent viewport’s world.

• void queue_screen_capture ( )

Queue a multithreaded screenshot, you can retrieve it at a later frame via **get_screen_capture**.

• void render_target_clear ( )

Clear the render target manually.

• void set_as_audio_listener ( **bool** enable )

Makes the viewport send sounds to the speakers.

• void set_as_audio_listener_2d ( **bool** enable )

Makes the viewport send sounds from 2D emitters to the speakers.

• void set_as_render_target ( **bool** enable )

Set the viewport’s render target mode.

• void set_canvas_transform ( **Matrix32** xform )

Set the canvas transform of the viewport, useful for changing the on-screen positions of all child **CanvasItems**. This is relative to the global canvas transform of the viewport.

• void set_disable_input ( **bool** disable )

Set whether input to the viewport is disabled.

• void set_global_canvas_transform ( **Matrix32** xform )

Set the global canvas transform of the viewport. The canvas transform is relative to this.

• void set_physics_object_picking ( **bool** enable )

Enable/disable picking for all physics objects inside the viewport.

• void set_rect ( **Rect2** rect )

Set the viewport rect. If the viewport is child of a control, it will use the same rect as the parent.

• void set_render_target_clear_on_new_frame ( **bool** enable )

Enable/disable automatic clearing of the render target on each frame. You might find it better to disable this if you are using the viewport for rarely updated textures. To clear manually, check **render_target_clear**.

• void set_render_target_filter ( **bool** enable )

Set whether the rendered texture should have filters enabled. Disable if you want the texture’s pixels be visible.

• void set_render_target_gen_mipmaps ( **bool** enable )

Set whether the rendered texture should have mipmaps generated. Mipmaps allow the texture to have better antialiasing from far away.

• void set_render_target_to_screen_rect ( **Rect2** rect )

Map a part of the screen to the render target directly.
• void set_render_target_update_mode (int mode)
Set when the render target should be updated, has to be one of the RENDER_TARGET_UPDATE* constants.

• void set_render_target_vflip (bool enable)
Set whether the render target should be flipped on the Y axis.

• void set_size_override (bool enable, Vector2 size=Vector2(-1, -1), Vector2 margin=Vector2(0, 0))
Set the size of the viewport. If the enable parameter is true, it would use the override, otherwise it would use the default size. If the size parameter is equal to (-1, -1), it won't update the size.

• void set_size_override_stretch (bool enabled)
Set whether the size override affects stretch as well.

• void set_transparent_background (bool enable)
If this viewport is a child of another viewport, keep the previously drawn background visible.

• void set_use_own_world (bool enable)
Make the viewport use a world separate from the parent viewport’s world.

• void set_world (World world)
Change the 3D world of the viewport.

• void set_world_2d (World2D world_2d)
• void unhandled_input (InputEvent local_event)
• void update_worlds ()
Force update of the 2D and 3D worlds.

• void warp_mouse (Vector2 to_pos)
Warp the mouse to a position, relative to the viewport.

12.361 ViewportSprite

Inherits: Node2D < CanvasItem < Node < Object

Category: Core

12.361.1 Brief Description
Displays a viewport as a sprite.
12.361.2 Member Functions

<table>
<thead>
<tr>
<th>Color</th>
<th>get_modulate ( ) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vector2</td>
<td>get_offset ( ) const</td>
</tr>
<tr>
<td>NodePath</td>
<td>get_viewport_path ( ) const</td>
</tr>
<tr>
<td>bool</td>
<td>is_centered ( ) const</td>
</tr>
<tr>
<td>void</td>
<td>set_centered ( bool centered )</td>
</tr>
<tr>
<td>void</td>
<td>set_modulate ( Color modulate )</td>
</tr>
<tr>
<td>void</td>
<td>set_offset ( Vector2 offset )</td>
</tr>
<tr>
<td>void</td>
<td>set_viewport_path ( NodePath path )</td>
</tr>
</tbody>
</table>

12.361.3 Description

Used to display a Viewport node at some position in the world, without having to mess with RenderTargetTextures.

12.361.4 Member Function Description

- **Color get_modulate ( ) const**
  Get color modulation for the texture. All texture pixels are multiplied by this color.

- **Vector2 get_offset ( ) const**
  get the offset to the origin of the texture.

- **NodePath get_viewport_path ( ) const**
  Return the path to the shown Viewport node.

- **bool is_centered ( ) const**
  Return whether the viewport’s texture is centered on the origin.

- **void set_centered ( bool centered )**
  Set whether the viewport’s texture should be centered on the origin.

- **void set_modulate ( Color modulate )**
  Set color modulation for the texture. All texture pixels are multiplied by this color. Color may contain rgb values above 1 to achieve a highlight effect.

- **void set_offset ( Vector2 offset )**
  Set the offset to the origin of the texture.

- **void set_viewport_path ( NodePath path )**
  Set the path to the shown Viewport node.

12.362 VisibilityEnabler

Inherits: VisibilityNotifier < Spatial < Node < Object

Category: Core
12.362.1 Brief Description
Enable certain nodes only when visible.

12.362.2 Member Functions

<table>
<thead>
<tr>
<th>bool</th>
<th>is_enabler_enabled ( int enabler ) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>set_enabler ( int enabler, bool enabled )</td>
</tr>
</tbody>
</table>

12.362.3 Numeric Constants

- **ENABLER_FREEZE_BODIES** = 1 — This enabler will freeze **RigidBody** nodes.
- **ENABLER_PAUSE_ANIMATIONS** = 0 — This enabler will pause **AnimationPlayer** nodes.
- **ENABLER_MAX** = 2

12.362.4 Description
The VisibilityEnabler will disable **RigidBody** and **AnimationPlayer** nodes when they are not visible. It will only affect other nodes within the same scene as the VisibilityEnabler itself.

12.362.5 Member Function Description

- **bool is_enabler_enabled ( int enabler ) const**
  Returns whether the specified enabler was set to true or not.
- **void set_enabler ( int enabler, bool enabled )**
  Set an enabler to true for all nodes of its type to be disabled when the VisibilityEnabler is not in view. See the constants for enablers and what they affect.

12.363 VisibilityEnabler2D

Inherits: **VisibilityNotifier2D** < **Node2D** < **CanvasItem** < **Node** < **Object**

Category: Core

12.363.1 Brief Description
Enable certain nodes only when visible.

12.363.2 Member Functions

<table>
<thead>
<tr>
<th>bool</th>
<th>is_enabler_enabled ( int enabler ) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>set_enabler ( int enabler, bool enabled )</td>
</tr>
</tbody>
</table>
12.363.3 Numeric Constants

- ENABLER_FREEZE_BODIES = 1 — This enabler will freeze RigidBody2D nodes.
- ENABLER_PAUSE_ANIMATIONS = 0 — This enabler will pause AnimationPlayer nodes.
- ENABLER_PAUSE_PARTICLES = 2 — This enabler will stop Particles2D nodes.
- ENABLER_PAUSE_ANIMATED_SPRITES = 5
- ENABLER_PARENT_PROCESS = 3 — This enabler will stop the parent’s _process function.
- ENABLER_PARENT_FIXED_PROCESS = 4 — This enabler will stop the parent’s _fixed_process function.
- ENABLER_MAX = 6

12.363.4 Description

The VisibilityEnabler2D will disable RigidBody2D, AnimationPlayer, and other nodes when they are not visible. It will only affect other nodes within the same scene as the VisibilityEnabler2D itself.

12.363.5 Member Function Description

- bool is_enabler_enabled ( int enabler ) const
  Returns whether the specified enabler was set to true or not.

- void set_enabler ( int enabler, bool enabled )
  Set an enabler to true for all nodes of its type to be disabled when the VisibilityEnabler2D is not in view. See the constants for enablers and what they affect.

12.364 VisibilityNotifier

Inherits: Spatial < Node < Object

Inherited By: VisibilityEnabler

Category: Core

12.364.1 Brief Description

Detect when the node is visible on screen.

12.364.2 Member Functions

<table>
<thead>
<tr>
<th>AABB</th>
<th>get_aabb ( ) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool</td>
<td>is_on_screen ( ) const</td>
</tr>
<tr>
<td>void</td>
<td>set_aabb ( AABB rect )</td>
</tr>
</tbody>
</table>
12.364.3 Signals

- `enter_camera (Object camera)`
  Emitted when the VisibilityNotifier enters a Camera's view.
- `enter_screen ()`
  Emitted when the VisibilityNotifier enters the screen.
- `exit_camera (Object camera)`
  Emitted when the VisibilityNotifier exits a Camera’s view.
- `exit_screen ()`
  Emitted when the VisibilityNotifier exits the screen.

12.364.4 Description

The VisibilityNotifier is used to notify when its bounding box enters the screen, is visible on the screen, or when it exits the screen.

12.364.5 Member Function Description

- `AABB get_aabb () const`
  Return the visibility bounding box of the VisibilityNotifier.
- `bool is_on_screen () const`
  Return true if any part of the bounding box is on the screen.
- `void set_aabb (AABB rect)`
  Set the visibility bounding box of the VisibilityNotifier.

12.365 VisibilityNotifier2D

Inherits: Node2D < CanvasItem < Node < Object

Inherited By: VisibilityEnabler2D

Category: Core

12.365.1 Brief Description

Detect when the node is visible on screen.

12.365.2 Member Functions

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rect2</td>
<td><code>get_rect () const</code></td>
<td>Return the visible bounding box of the node.</td>
</tr>
<tr>
<td>bool</td>
<td><code>is_on_screen () const</code></td>
<td>Return true if any part of the bounding box is on the screen.</td>
</tr>
<tr>
<td>void</td>
<td><code>set_rect (Rect2 rect)</code></td>
<td>Set the visibility bounding box of the node.</td>
</tr>
</tbody>
</table>
12.365.3 Signals

- **enter_screen ()**
  Emitted when the VisibilityNotifier2D enters the screen.

- **enter_viewport (Object viewport)**
  Emitted when the VisibilityNotifier2D enters a Viewport.

- **exit_screen ()**
  Emitted when the VisibilityNotifier2D exits the screen.

- **exit_viewport (Object viewport)**
  Emitted when the VisibilityNotifier2D exits a Viewport.

12.365.4 Description

The VisibilityNotifier2D is used to notify when its bounding rectangle enters the screen, is visible on the screen, or when it exits the screen.

12.365.5 Member Function Description

- **Rect2 get_rect () const**
  Return the visibility bounding rectangle of the VisibilityNotifier2D.

- **bool is_on_screen () const**
  Return true if any part of the bounding rectangle is on the screen.

- **void set_rect (Rect2 rect)**
  Set the visibility bounding rectangle of the VisibilityNotifier2D.

12.366 VisualInstance

**Inherits:** Spatial < Node < Object

**Inherited By:** BakedLightInstance, Light, Room, BakedLightSampler, Portal, GeometryInstance

**Category:** Core

12.366.1 Brief Description

12.366.2 Member Functions

<table>
<thead>
<tr>
<th>AABB</th>
<th>get_aabb () const</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>get_layer_mask () const</td>
</tr>
<tr>
<td>AABB</td>
<td>get_transformed_aabb () const</td>
</tr>
<tr>
<td>void</td>
<td>set_base (RID base)</td>
</tr>
<tr>
<td>void</td>
<td>set_layer_mask (int mask)</td>
</tr>
</tbody>
</table>
12.366.3 Member Function Description

- `AABB get_aabb ( ) const`
- `int get_layer_mask ( ) const`
- `AABB get_transformed_aabb ( ) const`
- `void set_base ( RID base )`
- `void set_layer_mask ( int mask )`

12.367 VisualServer

Inherits: `Object`
Category: Core

12.367.1 Brief Description

Server for anything visible.

12.367.2 Member Functions

```csharp
void black_bars_set_images ( RID left, RID top, RID right, RID bottom )
void black_bars_set_margins ( int left, int top, int right, int bottom )
RID camera_create ( )
void camera_set_orthogonal ( RID arg0, float arg1, float arg2, float arg3 )
void camera_set_perspective ( RID arg0, float arg1, float arg2, float arg3 )
void camera_set_transform ( RID arg0, Transform arg1 )
RID canvas_create ( )
void canvas_item_add_circle ( RID arg0, Vector2 arg1, float arg2, Color arg3 )
void canvas_item_add_line ( RID arg0, Vector2 arg1, Vector2 arg2, Color arg3, float arg4=1 )
void canvas_item_add_style_box ( RID arg0, Rect2 arg1, RID arg2, RealArray arg3, Color arg4=Color(1,1,1,1), bool arg5=false )
void canvas_item_add_texture_rect ( RID arg0, Rect2 arg1, RID arg2, bool arg3, Color arg4=Color(1,1,1,1), bool arg5=false )
void canvas_item_add_texture_rect_region ( RID arg0, Rect2 arg1, RID arg2, Rect2 arg3, Color arg4=Color(1,1,1,1), bool arg5=false )
void canvas_item_clear ( RID arg0 )
RID canvas_item_create ( )
float canvas_item_get_opacity ( RID arg0, float arg1 ) const
RID canvas_item_get_parent ( RID arg0 ) const
float canvas_item_get_self_opacity ( RID arg0, float arg1 ) const
void canvas_item_raise ( RID arg0 )
void canvas_item_set_clip ( RID arg0, bool arg1 )
void canvas_item_set_custom_rect ( RID arg0, bool arg1, Rect2 arg2 )
void canvas_item_set_opacity ( RID arg0, float arg1 )
void canvas_item_set_parent ( RID arg0, RID arg1 )
void canvas_item_set_self_opacity ( RID arg0, float arg1 )
void canvas_item_set_sort_children_by_y ( RID arg0, bool arg1 )
void canvas_item_set_transform ( RID arg0, Matrix32 arg1 )
```
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void canvas_item_set_z ( RID arg0, int arg1 )</code></td>
<td>canvas item set z</td>
</tr>
<tr>
<td><code>void cursor_set_pos ( Vector2 arg0, int arg1 )</code></td>
<td>cursor set pos</td>
</tr>
<tr>
<td><code>void cursor_set_rotation ( float arg0, int arg1 )</code></td>
<td>cursor set rotation</td>
</tr>
<tr>
<td><code>void cursor_set_texture ( RID arg0, Vector2 arg1, int arg2, Rect2 arg3 )</code></td>
<td>cursor set texture</td>
</tr>
<tr>
<td><code>void cursor_set_visible ( bool arg0, int arg1 )</code></td>
<td>cursor set visible</td>
</tr>
<tr>
<td><code>void draw ( )</code></td>
<td>draw</td>
</tr>
<tr>
<td><code>void free_rid ( RID arg0 )</code></td>
<td>free rid</td>
</tr>
<tr>
<td><code>Color get_default_clear_color ( ) const</code></td>
<td>get default clear color</td>
</tr>
<tr>
<td><code>int get_render_info ( int arg0 )</code></td>
<td>get render info</td>
</tr>
<tr>
<td><code>RID get_test_cube ( )</code></td>
<td>get test cube</td>
</tr>
<tr>
<td><code>void instance_attach_object_instance_ID ( RID arg0, int arg1 )</code></td>
<td>instance attach object instance ID</td>
</tr>
<tr>
<td><code>void instance_attach_skeleton ( RID arg0, RID arg1 )</code></td>
<td>instance attach skeleton</td>
</tr>
<tr>
<td><code>RID instance_create ( )</code></td>
<td>instance create</td>
</tr>
<tr>
<td><code>RID instance_geometry_get_material_param ( RID arg0 ) const</code></td>
<td>instance geometry get material param</td>
</tr>
<tr>
<td><code>RID instance_geometry_override_material_param ( RID arg0 ) const</code></td>
<td>instance geometry override material param</td>
</tr>
<tr>
<td><code>RID instance_get_base ( RID arg0 ) const</code></td>
<td>instance get base</td>
</tr>
<tr>
<td><code>RID instance_get_base_aabb ( RID arg0 ) const</code></td>
<td>instance get base aabb</td>
</tr>
<tr>
<td><code>int instance_get_instance_id ( RID arg0 ) const</code></td>
<td>instance get instance id</td>
</tr>
<tr>
<td><code>RID instance_get_room ( RID arg0 ) const</code></td>
<td>instance get room</td>
</tr>
<tr>
<td><code>RID instance_get_skeleton ( RID arg0 ) const</code></td>
<td>instance get skeleton</td>
</tr>
<tr>
<td><code>Transform instance_get_transform ( RID arg0 ) const</code></td>
<td>instance get transform</td>
</tr>
<tr>
<td><code>bool instance_is_exterior ( RID arg0 ) const</code></td>
<td>instance is exterior</td>
</tr>
<tr>
<td><code>void instance_set_exterior ( RID arg0, bool arg1 )</code></td>
<td>instance set exterior</td>
</tr>
<tr>
<td><code>void instance_set_room ( RID arg0, RID arg1 )</code></td>
<td>instance set room</td>
</tr>
<tr>
<td><code>Array instance_transform ( RID arg0, Transform arg1 )</code></td>
<td>instance transform</td>
</tr>
<tr>
<td><code>Array instances_cull_aabb ( AABB arg0, RID arg1 ) const</code></td>
<td>instances cull aabb</td>
</tr>
<tr>
<td><code>Array instances_cull_convex ( Array arg0, RID arg1 ) const</code></td>
<td>instances cull convex</td>
</tr>
<tr>
<td><code>Array instances_cull_ray ( Vector3 arg0, Vector3 arg1, RID arg2 ) const</code></td>
<td>instances cull ray</td>
</tr>
<tr>
<td><code>RID light_create ( int arg0 )</code></td>
<td>light create</td>
</tr>
<tr>
<td><code>Color light_get_color ( RID arg0, int arg1 ) const</code></td>
<td>light get color</td>
</tr>
<tr>
<td><code>RID light_get_projector ( RID arg0 ) const</code></td>
<td>light get projector</td>
</tr>
<tr>
<td><code>int light_get_type ( RID arg0 ) const</code></td>
<td>light get type</td>
</tr>
<tr>
<td><code>float light_get_var ( RID arg0, int arg1 ) const</code></td>
<td>light get var</td>
</tr>
<tr>
<td><code>bool light_has_shadow ( RID arg0 ) const</code></td>
<td>light has shadow</td>
</tr>
<tr>
<td><code>bool light_is_volumetric ( RID arg0 ) const</code></td>
<td>light is volumetric</td>
</tr>
<tr>
<td><code>void light_set_color ( RID arg0, int arg1, Color arg2 )</code></td>
<td>light set color</td>
</tr>
<tr>
<td><code>void light_set_projector ( RID arg0, RID arg1 )</code></td>
<td>light set projector</td>
</tr>
<tr>
<td><code>void light_set_shadow ( RID arg0, bool arg1 )</code></td>
<td>light set shadow</td>
</tr>
<tr>
<td><code>void light_set_var ( RID arg0, int arg1, float arg2 )</code></td>
<td>light set var</td>
</tr>
<tr>
<td><code>void light_set_volumetric ( RID arg0, bool arg1 )</code></td>
<td>light set volumetric</td>
</tr>
<tr>
<td><code>RID make_sphere_mesh ( int arg0, int arg1, float arg2 )</code></td>
<td>make sphere mesh</td>
</tr>
<tr>
<td><code>RID material_create ( )</code></td>
<td>material create</td>
</tr>
<tr>
<td><code>int material_get_blend_mode ( RID arg0 ) const</code></td>
<td>material get blend mode</td>
</tr>
<tr>
<td><code>bool material_get_flag ( RID arg0, int arg1 ) const</code></td>
<td>material get flag</td>
</tr>
<tr>
<td><code>float material_get_line_width ( RID arg0 ) const</code></td>
<td>material get line width</td>
</tr>
<tr>
<td><code>void material_get_param ( RID arg0, String arg1 ) const</code></td>
<td>material get param</td>
</tr>
<tr>
<td><code>RID material_get_shader ( RID arg0 ) const</code></td>
<td>material get shader</td>
</tr>
<tr>
<td><code>void material_set_blend_mode ( RID arg0, int arg1 )</code></td>
<td>material set blend mode</td>
</tr>
<tr>
<td><code>void material_set_flag ( RID arg0, int arg1, bool arg2 )</code></td>
<td>material set flag</td>
</tr>
<tr>
<td><code>void material_set_line_width ( RID arg0, float arg1 )</code></td>
<td>material set line width</td>
</tr>
</tbody>
</table>

Continued on next page
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>void material_set_param(RID arg0, String arg1, Variant arg2)</code></td>
<td>Set material parameter</td>
</tr>
<tr>
<td><code>void material_set_shader(RID shader, RID arg1)</code></td>
<td>Set material shader</td>
</tr>
<tr>
<td><code>void mesh_add_surface(RID arg0, int arg1, Array arg2, Array arg3=Array(), bool arg4=false)</code></td>
<td>Add mesh surface</td>
</tr>
<tr>
<td><code>void mesh_add_surface_from_planes(RID arg0, Array arg1)</code></td>
<td>Add mesh surface from planes</td>
</tr>
<tr>
<td><code>RID mesh_create()</code></td>
<td>Create mesh</td>
</tr>
<tr>
<td><code>int mesh_get_surface_count(RID arg0)</code></td>
<td>Get mesh surface count</td>
</tr>
<tr>
<td><code>void mesh_remove_surface(RID arg0, int arg1)</code></td>
<td>Remove mesh surface</td>
</tr>
<tr>
<td><code>int mesh_surface_get_array_index_len(RID arg0, int arg1)</code></td>
<td>Get mesh surface array index length</td>
</tr>
<tr>
<td><code>int mesh_surface_get_array_len(RID arg0, int arg1)</code></td>
<td>Get mesh surface array length</td>
</tr>
<tr>
<td><code>int mesh_surface_get_format(RID arg0, int arg1)</code></td>
<td>Get mesh surface format</td>
</tr>
<tr>
<td><code>color mesh_surface_get_material(RID arg0, int arg1)</code></td>
<td>Get mesh surface material</td>
</tr>
<tr>
<td><code>int mesh_surface_get_primitive_type(RID arg0, int arg1)</code></td>
<td>Get mesh surface primitive type</td>
</tr>
<tr>
<td><code>void mesh_get_surface_count(RID arg0)</code></td>
<td>Get mesh surface count</td>
</tr>
<tr>
<td><code>void mesh_surface_get_material(RID arg0)</code></td>
<td>Get mesh surface material</td>
</tr>
<tr>
<td><code>void mesh_surface_get_primitive_type(RID arg0)</code></td>
<td>Get mesh surface primitive type</td>
</tr>
<tr>
<td><code>void particles_set_color_phase_pos(RID arg0, int arg1)</code></td>
<td>Set particle color phase position</td>
</tr>
<tr>
<td><code>void particles_set_height_from_velocity(RID arg0)</code></td>
<td>Set particle height from velocity</td>
</tr>
<tr>
<td><code>void particles_set_material(RID arg0)</code></td>
<td>Set particle material</td>
</tr>
<tr>
<td><code>void particles_set_randomness(RID arg0)</code></td>
<td>Set particle randomness</td>
</tr>
<tr>
<td><code>void particles_set_variable(RID arg0, int arg1)</code></td>
<td>Set particle variable</td>
</tr>
<tr>
<td><code>void particles_has_height_from_velocity(RID arg0)</code></td>
<td>Has particle height from velocity</td>
</tr>
<tr>
<td><code>void particles_set_emitting(RID arg0)</code></td>
<td>Set particle emitting</td>
</tr>
<tr>
<td><code>void particles_set_attractor_strength(RID arg0, int arg1)</code></td>
<td>Set particle attractor strength</td>
</tr>
<tr>
<td><code>void particles_set_attractor_pos(RID arg0, int arg1)</code></td>
<td>Set particle attractor position</td>
</tr>
<tr>
<td><code>void portal_create()</code></td>
<td>Create portal</td>
</tr>
<tr>
<td><code>float portal_get_disable_distance(RID arg0)</code></td>
<td>Get portal disable distance</td>
</tr>
</tbody>
</table>

Continued on next page
Table 12.42 – continued from previous page

<table>
<thead>
<tr>
<th>Color</th>
<th>portal_get_disabled_color ( RID arg0 ) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vector2Array</td>
<td>portal_get_shape ( RID arg0 ) const</td>
</tr>
<tr>
<td>bool</td>
<td>portal_is_enabled ( RID arg0 ) const</td>
</tr>
<tr>
<td>void</td>
<td>portal_set_disable_distance ( RID arg0, float arg1 )</td>
</tr>
<tr>
<td>void</td>
<td>portal_set_disabled_color ( RID arg0, Color arg1 )</td>
</tr>
<tr>
<td>void</td>
<td>portal_set_enabled ( RID arg0, bool arg1 )</td>
</tr>
<tr>
<td>void</td>
<td>portal_set_shape ( RID arg0, Vector2Array arg1 )</td>
</tr>
<tr>
<td>RID</td>
<td>room_create ( )</td>
</tr>
<tr>
<td>Dictionary</td>
<td>room_get_bounds ( RID arg0 ) const</td>
</tr>
<tr>
<td>void</td>
<td>room_set_bounds ( RID arg0, Dictionary arg1 )</td>
</tr>
<tr>
<td>RID</td>
<td>scenario_create ( )</td>
</tr>
<tr>
<td>void</td>
<td>scenario_set_debug ( RID arg0, int arg1 )</td>
</tr>
<tr>
<td>void</td>
<td>set_default_clear_color ( Color arg0 )</td>
</tr>
<tr>
<td>void</td>
<td>set_time_scale ( float arg0 )</td>
</tr>
<tr>
<td>RID</td>
<td>shader_create ( int mode=0 )</td>
</tr>
<tr>
<td>void</td>
<td>shader_set_mode ( RID shader, int mode )</td>
</tr>
<tr>
<td>Transform</td>
<td>skeleton_bone_get_transform ( RID arg0, int arg1 )</td>
</tr>
<tr>
<td>void</td>
<td>skeleton_bone_set_transform ( RID arg0, int arg1, Transform arg2 )</td>
</tr>
<tr>
<td>RID</td>
<td>skeleton_create ( )</td>
</tr>
<tr>
<td>int</td>
<td>skeleton_get_bone_count ( RID arg0 ) const</td>
</tr>
<tr>
<td>void</td>
<td>skeleton_resize ( RID arg0, int arg1 )</td>
</tr>
<tr>
<td>void</td>
<td>sync ( )</td>
</tr>
<tr>
<td>RID</td>
<td>texture_create ( )</td>
</tr>
<tr>
<td>RID</td>
<td>texture_create_from_image ( Image arg0, int arg1=7 )</td>
</tr>
<tr>
<td>int</td>
<td>texture_get_flags ( RID arg0 ) const</td>
</tr>
<tr>
<td>int</td>
<td>texture_get_height ( RID arg0 ) const</td>
</tr>
<tr>
<td>int</td>
<td>texture_get_width ( RID arg0 ) const</td>
</tr>
<tr>
<td>void</td>
<td>texture_set_flags ( RID arg0, int arg1 )</td>
</tr>
<tr>
<td>void</td>
<td>texture_set_shrink_all_x2_on_set_data ( bool shrink )</td>
</tr>
<tr>
<td>void</td>
<td>viewport_attach_camera ( RID arg0, RID arg1=RID() )</td>
</tr>
<tr>
<td>void</td>
<td>viewport_attach_canvas ( RID arg0, RID arg1 )</td>
</tr>
<tr>
<td>RID</td>
<td>viewport_create ( )</td>
</tr>
<tr>
<td>RID</td>
<td>viewport_get_attached_camera ( RID arg0 ) const</td>
</tr>
<tr>
<td>Rect2</td>
<td>viewport_get_rect ( RID arg0 ) const</td>
</tr>
<tr>
<td>RID</td>
<td>viewport_remove_scenario ( RID arg0 ) const</td>
</tr>
<tr>
<td>void</td>
<td>viewport_remove_canvas ( RID arg0, RID arg1 )</td>
</tr>
<tr>
<td>void</td>
<td>viewport_set_canvas_transform ( RID arg0, RID arg1, Matrix32 arg2 )</td>
</tr>
<tr>
<td>void</td>
<td>viewport_set_global_canvas_transform ( RID arg0, Matrix32 arg1 )</td>
</tr>
<tr>
<td>void</td>
<td>viewport_set_rect ( RID arg0, Rect2 arg1 )</td>
</tr>
</tbody>
</table>

12.367.3 Numeric Constants

- **NO_INDEX_ARRAY** = -1
- **CUSTOM_ARRAY_SIZE** = 8
- **ARRAY_WEIGHTS_SIZE** = 4
- **MAX_PARTICLE_COLOR_PHASES** = 4
- **MAX_PARTICLE_ATTRACTORS** = 4
- **MAX_CURSORS** = 8
• `TEXTURE_FLAG_MIPMAPS = 1`
• `TEXTURE_FLAG_REPEAT = 2`
• `TEXTURE_FLAG_FILTER = 4`
• `TEXTURE_FLAG_CUBEMAP = 2048`
• `TEXTURE_FLAGS_DEFAULT = 7`
• `CUBEMAP_LEFT = 0`
• `CUBEMAP_RIGHT = 1`
• `CUBEMAP_BOTTOM = 2`
• `CUBEMAP_TOP = 3`
• `CUBEMAP_FRONT = 4`
• `CUBEMAP_BACK = 5`
• `SHADER_MATERIAL = 0`
• `SHADER_POST_PROCESS = 2`
• `MATERIAL_FLAG_VISIBLE = 0`
• `MATERIAL_FLAG_DOUBLE_SIDED = 1`
• `MATERIAL_FLAG_INVERT_FACES = 2`
• `MATERIAL_FLAG_UNSHADED = 3`
• `MATERIAL_FLAG_ONTOP = 4`
• `MATERIAL_FLAG_MAX = 7`
• `MATERIAL_BLEND_MODE_MIX = 0`
• `MATERIAL_BLEND_MODE_ADD = 1`
• `MATERIAL_BLEND_MODE_SUB = 2`
• `MATERIAL_BLEND_MODE_MUL = 3`
• `FIXED_MATERIAL_PARAM_DIFFUSE = 0`
• `FIXED_MATERIAL_PARAM_DETAIL = 1`
• `FIXED_MATERIAL_PARAM_SPECULAR = 2`
• `FIXED_MATERIAL_PARAM_EMISSION = 3`
• `FIXED_MATERIAL_PARAM_SPECULAR_EXP = 4`
• `FIXED_MATERIAL_PARAM_GLOW = 5`
• `FIXED_MATERIAL_PARAM_NORMAL = 6`
• `FIXED_MATERIAL_PARAM_SHADE_PARAM = 7`
• `FIXED_MATERIAL_PARAM_MAX = 8`
• `FIXED_MATERIAL_TEXCOORD_SPHERE = 3`
• `FIXED_MATERIAL_TEXCOORD_UV = 0`
• `FIXED_MATERIAL_TEXCOORD_UV_TRANSFORM = 1`
• `FIXED_MATERIAL_TEXCOORD_UV2 = 2`
• ARRAY_VERTEX = 0
• ARRAY_NORMAL = 1
• ARRAY_TANGENT = 2
• ARRAY_COLOR = 3
• ARRAY_TEX_UV = 4
• ARRAY_BONES = 6
• ARRAY_WEIGHTS = 7
• ARRAY_INDEX = 8
• ARRAY_MAX = 9
• ARRAY_FORMAT_VERTEX = 1
• ARRAY_FORMAT_NORMAL = 2
• ARRAY_FORMAT_TANGENT = 4
• ARRAY_FORMAT_COLOR = 8
• ARRAY_FORMAT_TEX_UV = 16
• ARRAY_FORMAT_BONES = 64
• ARRAY_FORMAT_WEIGHTS = 128
• ARRAY_FORMAT_INDEX = 256
• PRIMITIVE_POINTS = 0
• PRIMITIVE_LINES = 1
• PRIMITIVE_LINE_STRIP = 2
• PRIMITIVE_LINE_LOOP = 3
• PRIMITIVE_TRIANGLES = 4
• PRIMITIVE_TRIANGLE_STRIP = 5
• PRIMITIVE_TRIANGLE_FAN = 6
• PRIMITIVE_MAX = 7
• PARTICLE_LIFETIME = 0
• PARTICLE_SPREAD = 1
• PARTICLE_GRAVITY = 2
• PARTICLE_LINEAR_VELOCITY = 3
• PARTICLE_ANGULAR_VELOCITY = 4
• PARTICLE_LINEAR_ACCELERATION = 5
• PARTICLE_RADIAL_ACCELERATION = 6
• PARTICLE_TANGENTIAL_ACCELERATION = 7
• PARTICLE_INITIAL_SIZE = 9
• PARTICLE_FINAL_SIZE = 10
• PARTICLE_INITIAL_ANGLE = 11
• PARTICLE_HEIGHT = 12
• PARTICLE_HEIGHT_SPEED_SCALE = 13
• PARTICLE_VAR_MAX = 14
• LIGHT_DIRECTIONAL = 0
• LIGHT_OMNI = 1
• LIGHT_SPOT = 2
• LIGHT_COLOR_DIFFUSE = 0
• LIGHT_COLOR_SPECULAR = 1
• LIGHT_PARAM_SPOT_ATTENUATION = 0
• LIGHT_PARAM_SPOT_ANGLE = 1
• LIGHT_PARAM_RADIUS = 2
• LIGHT_PARAM_ENERGY = 3
• LIGHT_PARAM_ATTENUATION = 4
• LIGHT_PARAM_MAX = 10
• SCENARIO_DEBUG_DISABLED = 0
• SCENARIO_DEBUG_WIREFRAME = 1
• SCENARIO_DEBUG_OVERDRAW = 2
• INSTANCE_MESH = 1
• INSTANCE_MULTIMESH = 2
• INSTANCE_PARTICLES = 4
• INSTANCE_LIGHT = 5
• INSTANCE_ROOM = 6
• INSTANCE_PORTAL = 7
• INSTANCE_GEOMETRY_MASK = 30
• INFO_OBJECTS_IN_FRAME = 0
• INFO_VERTICES_IN_FRAME = 1
• INFO_MATERIAL_CHANGES_IN_FRAME = 2
• INFO_SHADER_CHANGES_IN_FRAME = 3
• INFO_SURFACE_CHANGES_IN_FRAME = 4
• INFO_DRAW_CALLS_IN_FRAME = 5
• INFO_USAGE_VIDEO_MEM_TOTAL = 6
• INFO_VIDEO_MEM_USED = 7
• INFO_TEXTURE_MEM_USED = 8
• INFO_VERTEX_MEM_USED = 9
12.367.4 Description

Server for anything visible. The visual server is the API backend for everything visible. The whole scene system mounts on it to display.

The visual server is completely opaque, the internals are entirely implementation specific and cannot be accessed.

12.367.5 Member Function Description

- void black_bars_set_images (RID left, RID top, RID right, RID bottom)
- void black_bars_set_margins (int left, int top, int right, int bottom)
- RID camera_create()
- void camera_set_orthogonal (RID arg0, float arg1, float arg2, float arg3)
- void camera_set_perspective (RID arg0, float arg1, float arg2, float arg3)
- void camera_set_transform (RID arg0, Transform arg1)
- RID canvas_create()
- void canvas_item_add_circle (RID arg0, Vector2 arg1, float arg2, Color arg3)
- void canvas_item_add_line (RID arg0, Vector2 arg1, Vector2 arg2, Color arg3, float arg4=1)
- void canvas_item_add_rect (RID arg0, Rect2 arg1, Color arg2)
- void canvas_item_add_style_box (RID arg0, Rect2 arg1, Rect2 arg2, RID arg3, RealArray arg4, Color arg5=Color(1,1,1,1), bool arg5=false)
- void canvas_item_add_texture_rect (RID arg0, Rect2 arg1, RID arg2, bool arg3, Color arg4=Color(1,1,1,1), bool arg5=false)
- void canvas_item_add_texture_rect_region (RID arg0, Rect2 arg1, RID arg2, Rect2 arg3, Color arg4=Color(1,1,1,1), bool arg5=false)
- void canvas_item_clear (RID arg0)
- RID canvas_item_create()
- float canvas_item_get_opacity (RID arg0, float arg1) const
- RID canvas_item_get_parent (RID arg0) const
- float canvas_item_get_self_opacity (RID arg0, float arg1) const
- void canvas_item_raise (RID arg0)
- void canvas_item_set_clip (RID arg0, bool arg1)
- void canvas_item_set_custom_rect (RID arg0, bool arg1, Rect2 arg2)
- void canvas_item_set_opacity (RID arg0, float arg1)
- void canvas_item_set_parent (RID arg0, RID arg1)
- void canvas_item_set_self_opacity (RID arg0, float arg1)
- void canvas_item_set_sort_children_by_y (RID arg0, bool arg1)
- void canvas_item_set_transform (RID arg0, Matrix32 arg1)
- void canvas_item_set_z (RID arg0, int arg1)
- void cursor_set_pos (Vector2 arg0, int arg1)
• void cursor_set_rotation ( float arg0, int arg1 )
• void cursor_set_texture ( RID arg0, Vector2 arg1, int arg2, Rect2 arg3 )
• void cursor_set_visible ( bool arg0, int arg1 )
• void draw ()
• void free_rid ( RID arg0 )
• Color get_default_clear_color ( ) const
• int get_render_info ( int arg0 )
• RID get_test_cube ( )
• void instance_attach_object_instance_ID ( RID arg0, int arg1 )
• void instance_attach_skeleton ( RID arg0, RID arg1 )
• RID instance_create ( )
• RID instance_geometry_get_material_param ( RID arg0 ) const
• RID instance_geometry_override_material_param ( RID arg0 ) const
• RID instance_get_base ( RID arg0 ) const
• RID instance_get_base_aabb ( RID arg0 ) const
• int instance_get_object_instance_ID ( RID arg0 ) const
• RID instance_get_room ( RID arg0 ) const
• RID instance_get_skeleton ( RID arg0 ) const
• Transform instance_get_transform ( RID arg0 ) const
• bool instance_is_exterior ( RID arg0 ) const
• void instance_set_exterior ( RID arg0, bool arg1 )
• void instance_set_room ( RID arg0, RID arg1 )
• void instance_set_transform ( RID arg0, Transform arg1 )
• Array instances_cull_aabb ( AABB arg0, RID arg1 ) const
• Array instances_cull_convex ( Array arg0, RID arg1 ) const
• Array instances_cull_ray ( Vector3 arg0, Vector3 arg1, RID arg2 ) const
• RID light_create ( int arg0 )
• Color light_get_color ( RID arg0, int arg1 ) const
• RID light_get_projector ( RID arg0 ) const
• int light_get_type ( RID arg0 ) const
• float light_get_var ( RID arg0, int arg1 ) const
• bool light_has_shadow ( RID arg0 ) const
• bool light_is_volumetric ( RID arg0 ) const
• void light_set_color ( RID arg0, int arg1, Color arg2 )
• void light_set_projector ( RID arg0, RID arg1 )
• void light_set_shadow ( RID arg0, bool arg1 )
- `void light_set_var (RID arg0, int arg1, float arg2)`
- `void light_set_volumetric (RID arg0, bool arg1)`
- `RID make_sphere_mesh (int arg0, int arg1, float arg2)`
- `RID material_create ()`
- `int material_get_blend_mode (RID arg0)` const
- `bool material_get_flag (RID arg0, int arg1)` const
- `float material_get_line_width (RID arg0)` const
- `float material_get_param (RID arg0, String arg1)` const
- `RID material_get_shader (RID arg0)` const
- `void material_set_blend_mode (RID arg0, int arg1)`
- `void material_set_flag (RID arg0, int arg1, bool arg2)`
- `void material_set_line_width (RID arg0, float arg1)`
- `void material_set_param (RID arg0, String arg1, Variant arg2)`
- `void material_set_shader (RID shader, RID arg1)`
- `void mesh_add_surface (RID arg0, int arg1, Array arg2, Array arg3=Array(), bool arg4=false)`
- `void mesh_add_surface_from_planes (RID arg0, Array arg1)`
- `RID mesh_create ()`
- `int mesh_get_surface_count (RID arg0)` const
- `void mesh_remove_surface (RID arg0, int arg1)`
- `int mesh_surface_get_array_index_len (RID arg0, int arg1)` const
- `int mesh_surface_get_array_len (RID arg0, int arg1)` const
- `int mesh_surface_get_format (RID arg0, int arg1)` const
- `RID mesh_surface_get_material (RID arg0, int arg1)` const
- `int mesh_surface_get_primitive_type (RID arg0, int arg1)` const
- `void mesh_surface_set_material (RID arg0, int arg1, RID arg2, bool arg3=false)`
- `RID multimesh_create ()`
- `AABB multimesh_get_aabb (RID arg0, AABB arg1)` const
- `RID multimesh_get_mesh (RID arg0)` const
- `Color multimesh_instance_get_color (RID arg0, int arg1)` const
- `Transform multimesh_instance_get_transform (RID arg0, int arg1)` const
- `void multimesh_instance_set_color (RID arg0, int arg1, Color arg2)`
- `void multimesh_instance_set_transform (RID arg0, int arg1, Transform arg2)`
- `void multimesh_set_aabb (RID arg0, AABB arg1)`
- `void multimesh_set_mesh (RID arg0, RID arg1)`
- `RID particles_create ()`
- `int particles_get_amount (RID arg0)` const
- `Vector3` `particles_get_attractor_pos ( RID arg0, int arg1 )` const
- `float` `particles_get_attractor_strength ( RID arg0, int arg1 )` const
- `int` `particles_get_attractors ( RID arg0 )` const
- `Color` `particles_get_color_phase_color ( RID arg0, int arg1 )` const
- `float` `particles_get_color_phase_pos ( RID arg0, int arg1 )` const
- `int` `particles_get_color_phases ( RID arg0 )` const
- `float` `particles_get_randomness ( RID arg0, int arg1 )` const
- `float` `particles_get_variable ( RID arg0, int arg1 )` const
- `AABB` `particles_get_visibility_aabb ( RID arg0 )` const
- `bool` `particles_has_height_from_velocity ( RID arg0 )` const
- `bool` `particles_is_emitting ( RID arg0 )` const
- `void` `particles_set_amount ( RID arg0, int arg1 )`
- `void` `particles_set_attractor_pos ( RID arg0, int arg1, Vector3 arg2 )`
- `void` `particles_set_attractor_strength ( RID arg0, int arg1, float arg2 )`
- `void` `particles_set_attractors ( RID arg0, int arg1 )`
- `void` `particles_set_color_phase_color ( RID arg0, int arg1, Color arg2 )`
- `void` `particles_set_color_phase_pos ( RID arg0, int arg1, float arg2 )`
- `void` `particles_set_color_phases ( RID arg0, int arg1 )`
- `void` `particles_set_emitting ( RID arg0, bool arg1 )`
- `void` `particles_set_height_from_velocity ( RID arg0, bool arg1 )`
- `void` `particles_set_material ( RID arg0, RID arg1, bool arg2=false )`
- `void` `particles_set_randomness ( RID arg0, int arg1, float arg2 )`
- `void` `particles_set_variable ( RID arg0, int arg1, float arg2 )`
- `void` `particles_set_visibility_aabb ( RID arg0, AABB arg1 )`
- `RID` `portal_create ( )`
- `float` `portal_get_disable_distance ( RID arg0 )` const
- `Color` `portal_get_disabled_color ( RID arg0 )` const
- `Vector2Array` `portal_get_shape ( RID arg0 )` const
- `bool` `portal_is_enabled ( RID arg0 )` const
- `void` `portal_set_disable_distance ( RID arg0, float arg1 )`
- `void` `portal_set_disabled_color ( RID arg0, Color arg1 )`
- `void` `portal_set_enabled ( RID arg0, bool arg1 )`
- `void` `portal_set_shape ( RID arg0, Vector2Array arg1 )`
- `RID` `room_create ( )`
- `Dictionary` `room_get_bounds ( RID arg0 )` const
- `void` `room_set_bounds ( RID arg0, Dictionary arg1 )`
- `RID` `scenario_create()`
- `void` `scenario_set_debug`(`RID` `arg0`, `int` `arg1`)
- `void` `set_default_clear_color`(`Color` `arg0`)
- `void` `set_time_scale`(`float` `arg0`)
- `RID` `shader_create`(`int` `mode=0`)
- `void` `shader_set_mode`(`RID` `shader`, `int` `mode`)
- `Transform` `skeleton_bone_get_transform`(`RID` `arg0`, `int` `arg1`)
- `void` `skeleton_bone_set_transform`(`RID` `arg0`, `int` `arg1`, `Transform` `arg2`)
- `RID` `skeleton_create()`
- `int` `skeleton_get_bone_count`(`RID` `arg0`) const
- `void` `skeleton_resize`(`RID` `arg0`, `int` `arg1`)
- `void` `sync()`
- `RID` `texture_create()`
- `RID` `texture_create_from_image`(`Image` `arg0`, `int` `arg1=7`)
- `int` `texture_get_flags`(`RID` `arg0`) const
- `int` `texture_get_height`(`RID` `arg0`) const
- `int` `texture_get_width`(`RID` `arg0`) const
- `void` `texture_set_flags`(`RID` `arg0`, `int` `arg1`)
- `void` `texture_set_shrink_all_x2_on_set_data`(`bool` `shrink`)
- `void` `viewport_attach_camera`(`RID` `arg0`, `RID` `arg1=RID()`)
- `void` `viewport_attach_canvas`(`RID` `arg0`, `RID` `arg1`)
- `RID` `viewport_create()`
- `RID` `viewport_get_attached_camera`(`RID` `arg0`) const
- `Rect2` `viewport_get_rect`(`RID` `arg0`) const
- `RID` `viewport_get_scenario`(`RID` `arg0`) const
- `void` `viewport_remove_canvas`(`RID` `arg0`, `RID` `arg1`)
- `void` `viewport_set_canvas_transform`(`RID` `arg0`, `RID` `arg1`, `Matrix32` `arg2`)
- `void` `viewport_set_global_canvas_transform`(`RID` `arg0`, `Matrix32` `arg1`)
- `void` `viewport_set_rect`(`RID` `arg0`, `Rect2` `arg1`)

### 12.368 VScrollBar

**Inherits:** `ScrollBar` < `Range` < `Control` < `CanvasItem` < `Node` < `Object`

**Category:** Core
12.368 1 Brief Description
Vertical version of ScrollBar, which goes from left (min) to right (max).

12.369 VSeparator

Inherits: Separator < Control < CanvasItem < Node < Object
Category: Core

12.369.1 Brief Description
Vertical version of Separator.

12.369.2 Description
Vertical version of Separator. It is used to separate objects horizontally, though (but it looks vertical!).

12.370 VSlider

Inherits: Slider < Range < Control < CanvasItem < Node < Object
Category: Core

12.370.1 Brief Description
Vertical slider.

12.370.2 Description
Vertical slider. See Slider. This one goes from left (min) to right (max).

12.371 VSplitContainer

Inherits: SplitContainer < Container < Control < CanvasItem < Node < Object
Category: Core

12.371.1 Brief Description
Vertical split container.

12.371.2 Description
Vertical split container. See SplitContainer. This goes from left to right.
12.372 WeakRef

Inherits: Reference < Object
Category: Core

12.372.1 Brief Description

Holds an Object, but does not contribute to the reference count if the object is a reference.

12.372.2 Member Functions

| Object | get_ref () const |

12.372.3 Description

A weakref can hold a Reference, without contributing to the reference counter. A weakref can be created from an Object using @GDScript.weakref. If this object is not a reference, weakref still works, however, it does not have any effect on the object. Weakrefs are useful in cases where multiple classes have variables that refer to each other. Without weakrefs, using these classes could lead to memory leaks, since both references keep each other from being released. Making part of the variables a weakref can prevent this cyclic dependency, and allows the references to be released.

12.372.4 Member Function Description

- Object get_ref () const
  
  Returns the Object this weakref is referring to.

12.373 WindowDialog

Inherits: Popup < Control < CanvasItem < Node < Object
Inherited By: AcceptDialog
Category: Core

12.373.1 Brief Description

Base class for window dialogs.

12.373.2 Member Functions

<table>
<thead>
<tr>
<th>TextureButton</th>
<th>get_close_button ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>get_title () const</td>
</tr>
<tr>
<td>void</td>
<td>set_title ( String title )</td>
</tr>
</tbody>
</table>
12.373.3 Description

Windowdialog is the base class for all window-based dialogs. It’s a by-default toplevel Control that draws a window decoration and allows motion and resizing.

12.373.4 Member Function Description

- `TextureButton get_close_button ( )`
  Return the close TextureButton.

- `String get_title ( ) const`
  Return the title of the window.

- `void set_title ( String title )`
  Set the title of the window.

12.374 World

Inherits: Resource < Reference < Object

Category: Core

12.374.1 Brief Description

Class that has everything pertaining to a world.

12.374.2 Member Functions

<table>
<thead>
<tr>
<th>Class</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>PhysicsDirectSpaceState</td>
<td>get_direct_space_state ( )</td>
</tr>
<tr>
<td>Environment</td>
<td>get_environment ( ) const</td>
</tr>
<tr>
<td>RID</td>
<td>get_scenario ( ) const</td>
</tr>
<tr>
<td>RID</td>
<td>get_sound_space ( ) const</td>
</tr>
<tr>
<td>void</td>
<td>set_environment ( Environment env )</td>
</tr>
</tbody>
</table>

12.374.3 Description

Class that has everything pertaining to a world. A physics space, a visual scenario and a sound space. Spatial nodes register their resources into the current world.

12.374.4 Member Function Description

- `PhysicsDirectSpaceState get_direct_space_state ( )`
- `Environment get_environment ( ) const`
- `RID get_scenario ( ) const`
- `RID get_sound_space ( ) const`
• RID get_space ( ) const
• void set_environment ( Environment env )

12.375 World2D

Inherits: Resource < Reference < Object

Category: Core

12.375.1 Brief Description

Class that has everything pertaining to a 2D world.

12.375.2 Member Functions

<table>
<thead>
<tr>
<th>Class</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>RID</td>
<td>get_canvas ( )</td>
</tr>
<tr>
<td>Physics2DDirectSpaceState</td>
<td>get_direct_space_state ( )</td>
</tr>
<tr>
<td>RID</td>
<td>get_sound_space ( )</td>
</tr>
<tr>
<td>RID</td>
<td>get_space ( )</td>
</tr>
</tbody>
</table>

12.375.3 Description

Class that has everything pertaining to a 2D world. A physics space, a visual scenario and a sound space. 2D nodes register their resources into the current 2D world.

12.375.4 Member Function Description

• RID get_canvas ( )
Retrieve the RID of this world’s canvas resource. Used by the VisualServer for 2D drawing.

• Physics2DDirectSpaceState get_direct_space_state ( )
Retrieve the state of this world’s physics space. This allows arbitrary querying for collision.

• RID get_sound_space ( )
Retrieve the RID of this world’s sound space resource. Used by the SpatialSound2DServer for 2D spatial audio.

• RID get_space ( )
Retrieve the RID of this world’s physics space resource. Used by the Physics2DServer for 2D physics, treating it as both a space and an area.

12.376 WorldEnvironment

Inherits: Spatial < Node < Object

Category: Core
12.376.1 Brief Description

Sets environment properties for the entire scene

12.376.2 Member Functions

<table>
<thead>
<tr>
<th>Environment</th>
<th>get_environment ( ) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>set_environment ( Environment env )</td>
</tr>
</tbody>
</table>

12.376.3 Description

The *WorldEnvironment* node can be added to a scene in order to set default *Environment* variables for the scene. The *WorldEnvironment* can be overridden by an *Environment* node set on the current *Camera*. Additionally, only one *WorldEnvironment* may be instanced in a given scene at a time. The *WorldEnvironment* allows the user to specify default lighting parameters (e.g. ambient lighting), various post-processing effects (e.g. SSAO, DOF, Tonemapping), and how to draw the background (e.g. solid color, skybox).

12.376.4 Member Function Description

- *Environment get_environment ( ) const*
  Return the *Environment* currently bound.
- *void set_environment ( Environment env )*
  Set the currently bound *Environment* to the one specified.

12.377 XMLParser

*Inherits: Reference < Object*

*Category: Core*

12.377.1 Brief Description

Low-level class for creating parsers for XML files.
12.377.2 Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>int get_attribute_count ( ) const</code></td>
<td>Get the amount of attributes in the current element.</td>
</tr>
<tr>
<td><code>String get_attribute_name ( int idx ) const</code></td>
<td>Get the name of the attribute specified by the index in <code>idx</code> argument.</td>
</tr>
<tr>
<td><code>String get_attribute_value ( int idx ) const</code></td>
<td>Get the value of the attribute specified by the index in <code>idx</code> argument.</td>
</tr>
<tr>
<td><code>int get_current_line ( ) const</code></td>
<td></td>
</tr>
<tr>
<td><code>String get_named_attribute_value ( String name ) const</code></td>
<td></td>
</tr>
<tr>
<td><code>String get_named_attribute_value_safe ( String name ) const</code></td>
<td></td>
</tr>
<tr>
<td><code>String get_node_data ( ) const</code></td>
<td></td>
</tr>
<tr>
<td><code>String get_node_name ( ) const</code></td>
<td></td>
</tr>
<tr>
<td><code>int get_node_offset ( ) const</code></td>
<td></td>
</tr>
<tr>
<td><code>int get_node_type ( )</code></td>
<td></td>
</tr>
<tr>
<td><code>bool has_attribute ( String name ) const</code></td>
<td></td>
</tr>
<tr>
<td><code>bool is_empty ( ) const</code></td>
<td></td>
</tr>
<tr>
<td><code>int open ( String file )</code></td>
<td></td>
</tr>
<tr>
<td><code>int open_buffer ( RawArray buffer )</code></td>
<td></td>
</tr>
<tr>
<td><code>int read ( )</code></td>
<td></td>
</tr>
<tr>
<td><code>int seek ( int pos )</code></td>
<td></td>
</tr>
<tr>
<td><code>void skip_section ( )</code></td>
<td></td>
</tr>
</tbody>
</table>

12.377.3 Numeric Constants

- NODE_NONE = 0 — There’s no node (no file or buffer opened)
- NODE_ELEMENT = 1 — Element (tag)
- NODE_ELEMENT_END = 2 — End of element
- NODE_TEXT = 3 — Text node
- NODE_COMMENT = 4 — Comment node
- NODE_CDATA = 5 — CDATA content
- NODE_UNKNOWN = 6 — Unknown node

12.377.4 Description

This class can serve as base to make custom XML parsers. Since XML is a very flexible standard, this interface is low level so it can be applied to any possible schema.

12.377.5 Member Function Description

- `int get_attribute_count ( ) const`
  Get the amount of attributes in the current element.

- `String get_attribute_name ( int idx ) const`
  Get the name of the attribute specified by the index in `idx` argument.

- `String get_attribute_value ( int idx ) const`
  Get the value of the attribute specified by the index in `idx` argument.
Get the current line in the parsed file (currently not implemented).

- `String get_named_attribute_value ( String name ) const`

Get the value of a certain attribute of the current element by name. This will raise an error if the element has no such attribute.

- `String get_named_attribute_value_safe ( String name ) const`

Get the value of a certain attribute of the current element by name. This will return an empty `String` if the attribute is not found.

- `String get_node_data ( ) const`

Get the contents of a text node. This will raise an error in any other type of node.

- `String get_node_name ( ) const`

Get the name of the current element node. This will raise an error if the current node type is not `NODE_ELEMENT` nor `NODE_ELEMENT_END`.

- `int get_node_offset ( ) const`

Get the byte offset of the current node since the beginning of the file or buffer.

- `int get_node_type ( )`

Get the type of the current node. Compare with `NODE_*` constants.

- `bool has_attribute ( String name ) const`

Check whether or not the current element has a certain attribute.

- `bool is_empty ( ) const`

Check whether the current element is empty (this only works for completely empty tags, e.g. `<element>`).

- `int open ( String file )`

Open a XML file for parsing. This returns an error code.

- `int open_buffer ( RawArray buffer )`

Open a XML raw buffer for parsing. This returns an error code.

- `int read ( )`

Read the next node of the file. This returns an error code.

- `int seek ( int pos )`

Move the buffer cursor to a certain offset (since the beginning) and read the next node there. This returns an error code.

- `void skip_section ( )`

Skips the current section. If the node contains other elements, they will be ignored and the cursor will go to the closing of the current element.

### 12.378 YSort

**Inherits:** Node2D < CanvasItem < Node < Object

**Category:** Core
12.378.1 Brief Description

Sort all child nodes based on their Y positions.

12.378.2 Member Functions

<table>
<thead>
<tr>
<th>bool</th>
<th>is_sort_enabled ( ) const</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>set_sort_enabled ( bool enabled )</td>
</tr>
</tbody>
</table>

12.378.3 Description

Sort all child nodes based on their Y positions. The child node must inherit from CanvasItem for it to be sorted. Nodes that have a higher Y position will be drawn later, so they will appear on top of nodes that have a lower Y position.

12.378.4 Member Function Description

- bool is_sort_enabled ( ) const
  Returns true if the children nodes are being sorted.

- void set_sort_enabled ( bool enabled )
  Set whether the children nodes are sorted or not. (default true)