

---

# Godot Engine Documentation

*Release latest*

**Juan Linietsky, Ariel Manzur and the Godot community**

February 07, 2016



<b>1 Learning step by step</b>	<b>3</b>
<b>2 Engine</b>	<b>59</b>
<b>3 2D tutorials</b>	<b>87</b>
<b>4 3D tutorials</b>	<b>159</b>
<b>5 Networking</b>	<b>225</b>
<b>6 Editor plugins</b>	<b>231</b>
<b>7 Miscellaneous</b>	<b>233</b>
<b>8 Asset pipeline</b>	<b>271</b>
<b>9 Classes reference</b>	<b>307</b>
<b>10 Languages</b>	<b>791</b>
<b>11 Cheat sheets</b>	<b>825</b>
<b>12 Compiling</b>	<b>831</b>
<b>13 Advanced</b>	<b>853</b>
<b>14 Contributing</b>	<b>885</b>



The main documentation for the site is organized into a few sections:

- *Tutorials*
- *Reference*
- *Community*



---

## Learning step by step

---

### 1.1 Scenes and nodes

#### 1.1.1 Introduction

My games are exquisite.



Imagine for a second that you are not a game developer anymore. Instead, You are 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 in 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.

#### 1.1.2 Nodes

But let's go to 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.

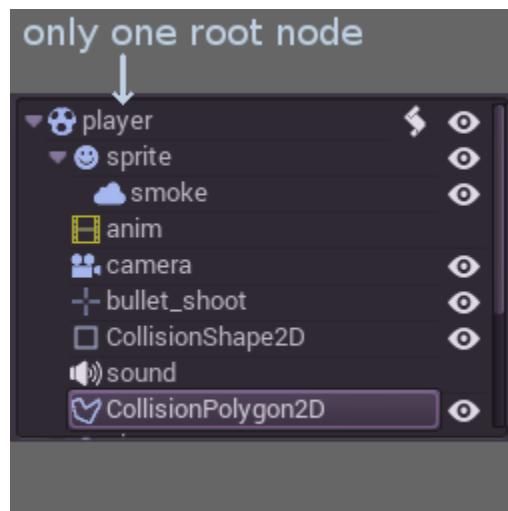
p=. **limage1**

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 the projects. Since different nodes have different functions, combining them allows to create more complex functions.

This is probably not clear yet and it 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.

### 1.1.3 Scenes



Now that the existence 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:

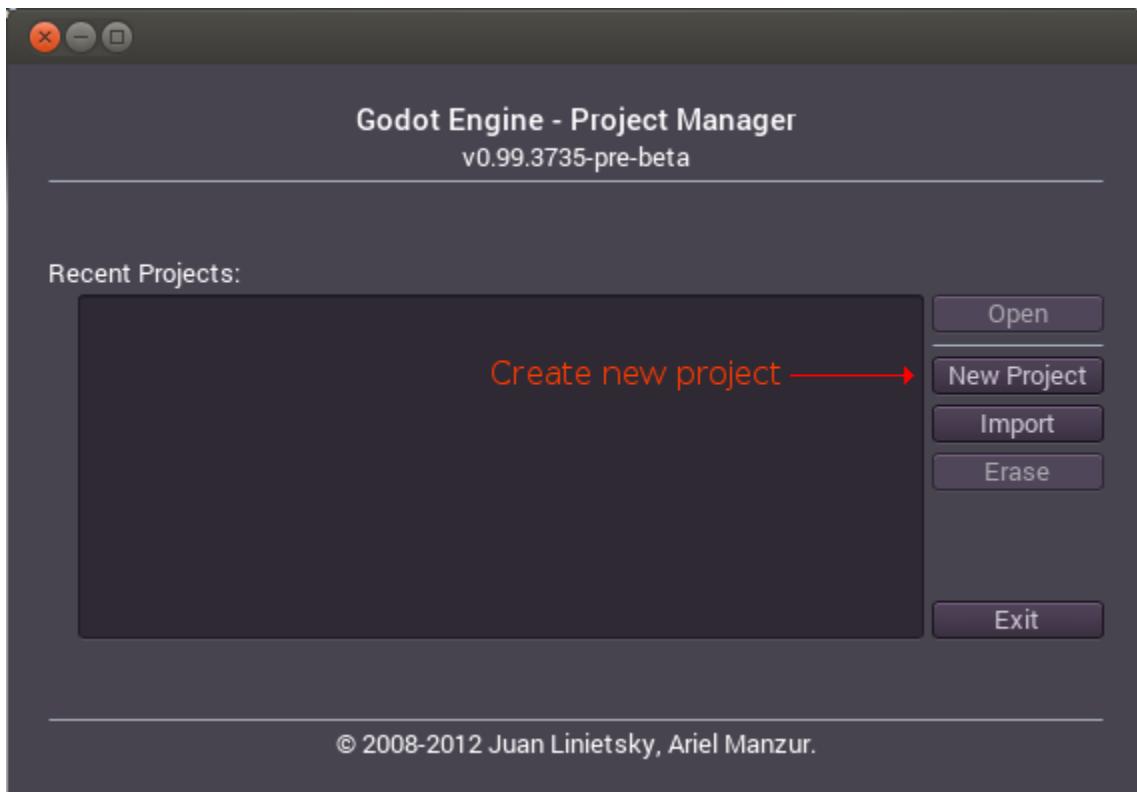
- A scene always has only one root node.
- Scenes can be saved to disk and loaded back.
- Scenes can be *instanced* (more on that later).
- Running a game means running a scene.
- There can be several scenes in a project, but for it to start, one of them must be selected to be loaded first.

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 all the editor revolves around the concept of editing a scene and the nodes that compose it.

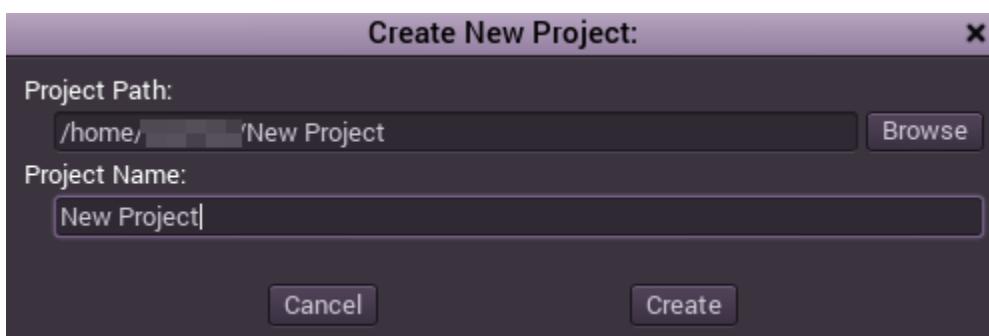
### 1.1.4 Creating New Project

Theory is boring, so let's change 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 godot executable is run outside a project, the Project Manager appears. This helps developers manage their projects.

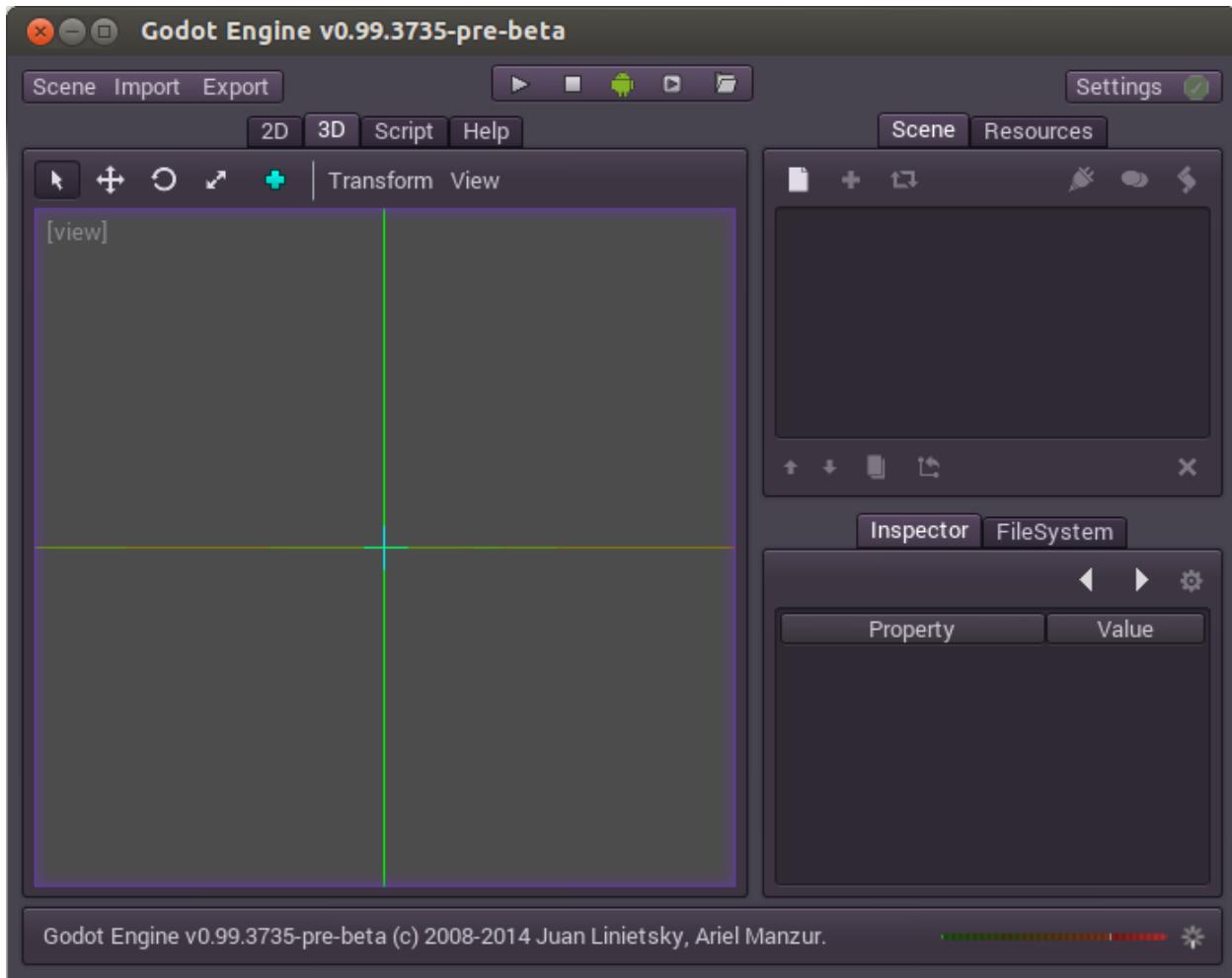


To create a new project, the “New Project” option must be used. Choose and create a path for the project and specify the project name:

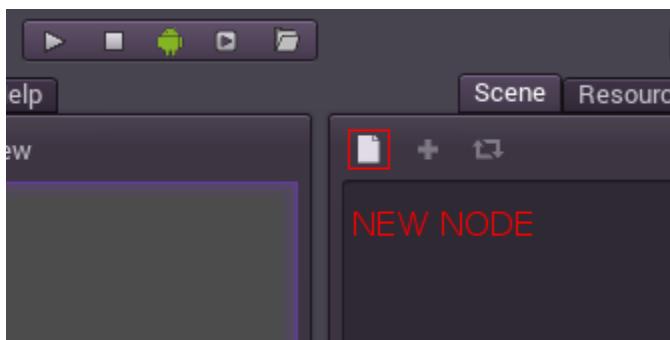


### 1.1.5 Editor

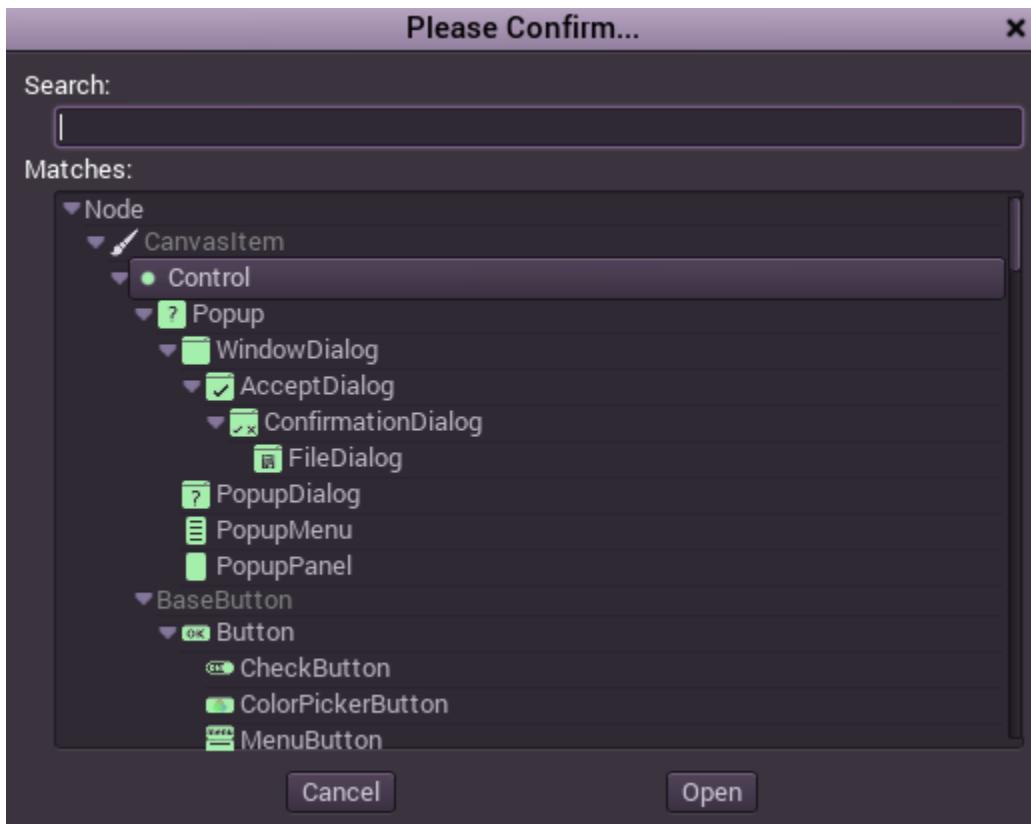
Once the “New Project” is created, the next step is opening it. This will open the Godot editor. Here is how the editor looks when freshly opened:



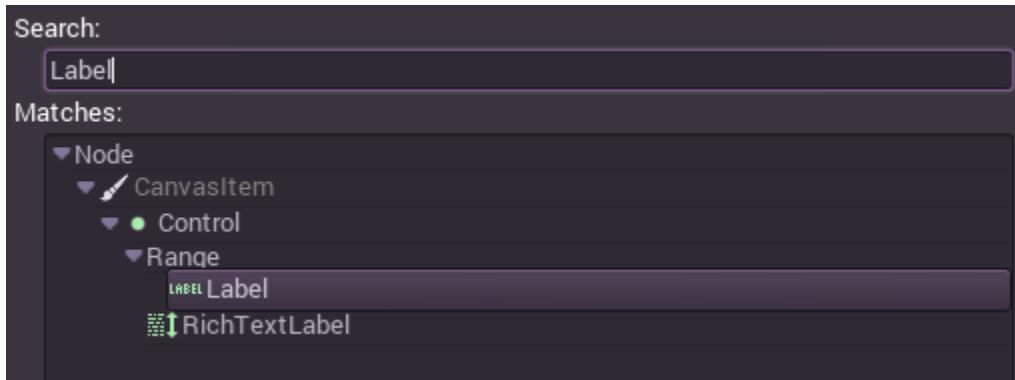
As mentioned before, making games in Godot feels like being in a kitchen, so let's open the refrigerator and add some fresh nodes to the project. We'll begin with a Hello World! To do this, the "New Node" button must be pressed:



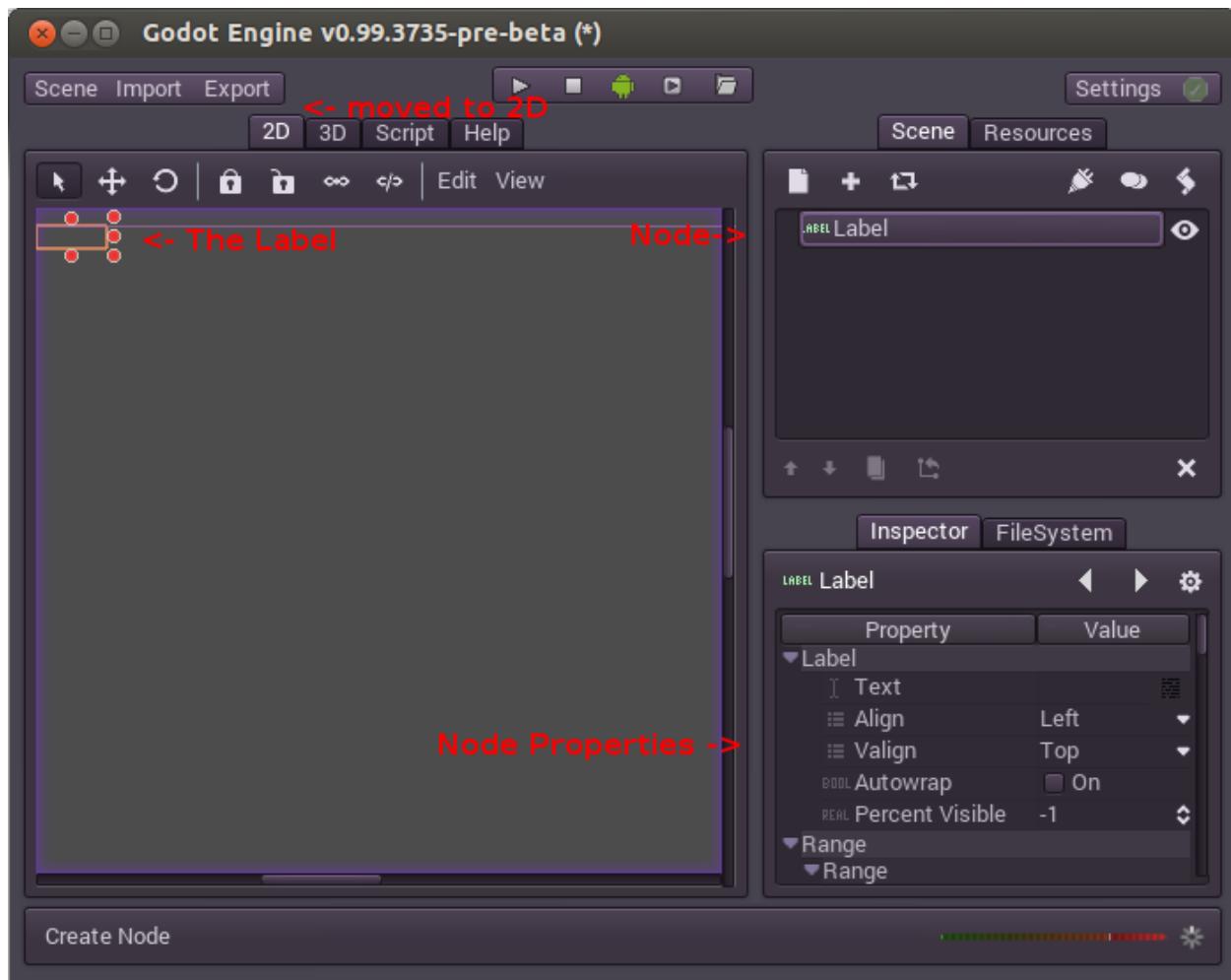
This will open the Create Node dialog, showing the long list of nodes that can be created:



From there, select the “Label” node first. Searching for it is probably the quickest way:



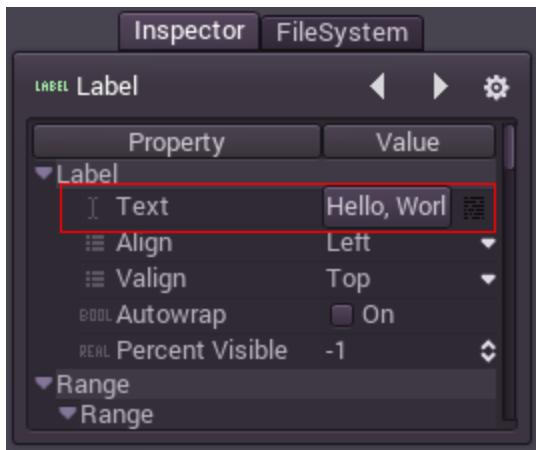
And finally, create the Label! A lot happens when Create is pressed:



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 right corner), and the label properties appear in the Inspector (box in the bottom right corner).

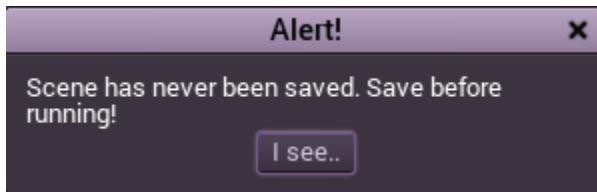
The next step, will be to change the “Text” Property of the label, let change it to “Hello, World!”:



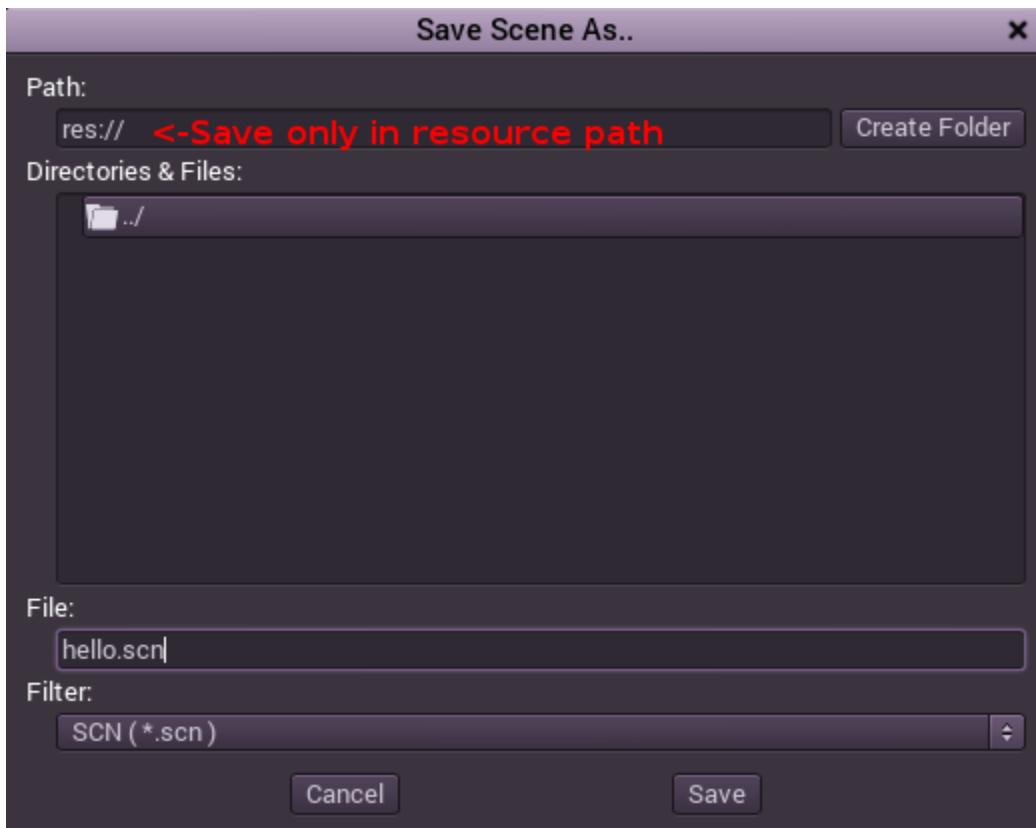
Ok, everything's ready to run the scene! Press the PLAY SCENE Button on the top bar (or hit F6):



Aaaand.. Oops.

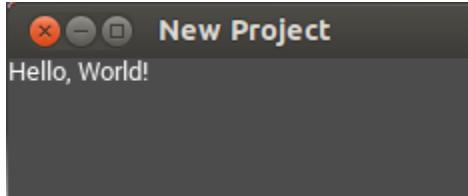


Scenes need to be saved to be run, so save the scene to something like hello.scn in Scene -> Save:



And here's when something funny happens. The file dialog is a special file dialog, and only allows 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

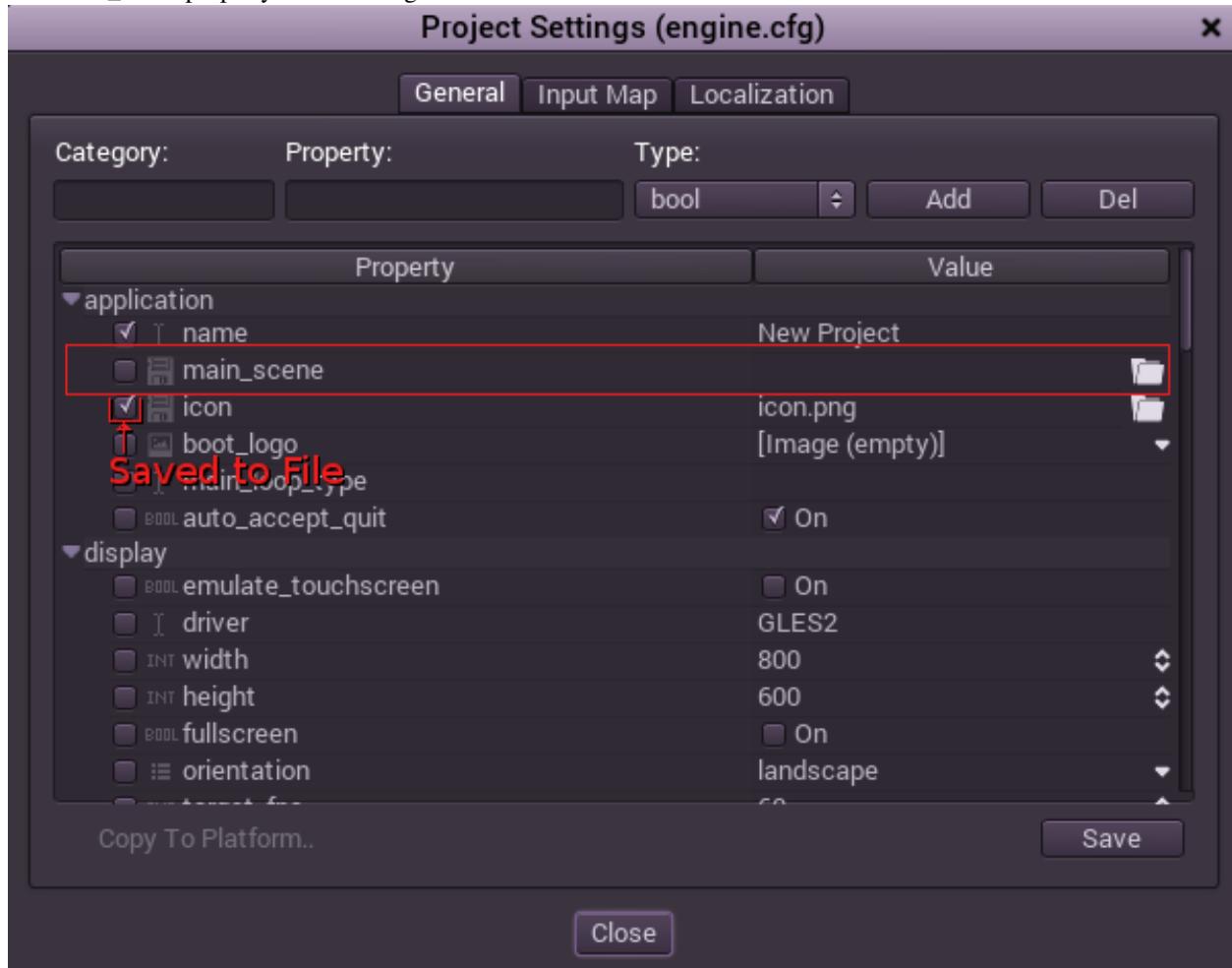
### 1.1.6 Configuring the Project

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 set in that 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 show 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 [Globals](#) singleton.

### 1.1.7 To Be Continued...

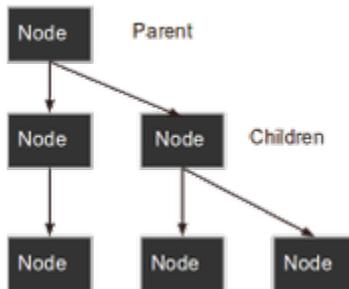
This tutorial talks about “Scenes and Nodes”, but so far there has been only *one* scene and *one* node! Don’t worry, the next tutorial will deal with that...

## 1.2 Instancing

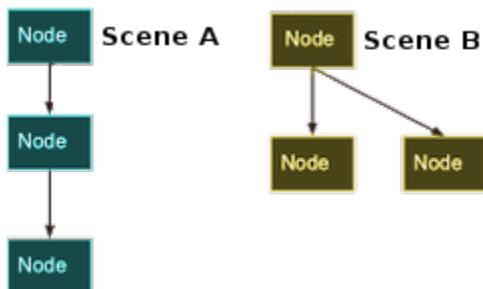
### 1.2.1 Rationale

Having a scene and throwing nodes to 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 in 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!

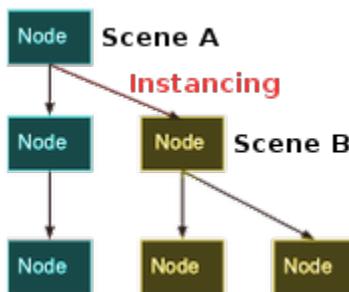
To recap: A scene is a collection of nodes organized as a tree, where they can have only one single node as the tree root.



In Godot, a scene can be created and saved it 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:

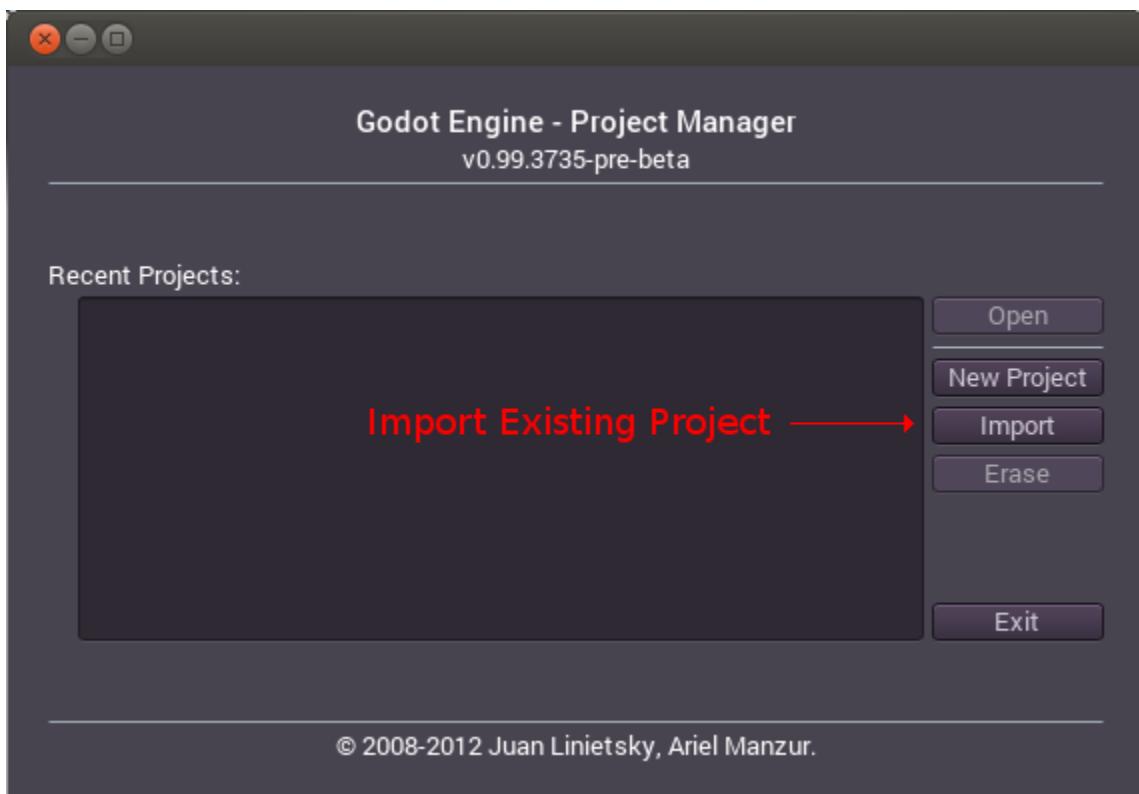


In the above picture, Scene B was added to Scene A as an instance. It may seem weird at first, but at the end of this tutorial it will make complete sense!

### 1.2.2 Instancing, Step by Step

To learn how to do instancing, let's start with downloading attachment:instancing.zip.

Unzip this scene in any place of our preference. Then, add this scene to the project manager using the 'Import' option:



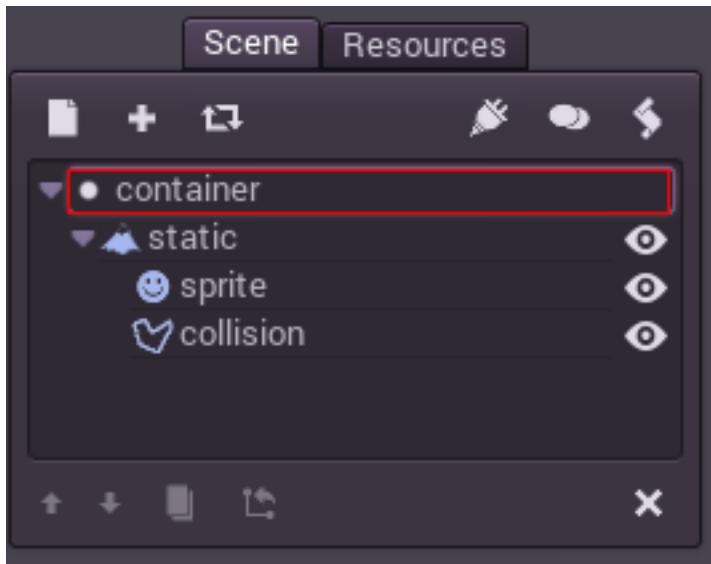
Simply browse to inside the project location and open the “engine.cfg” file. The new project will appear on the list of projects. Edit the project by using the ‘Edit’ option.

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.

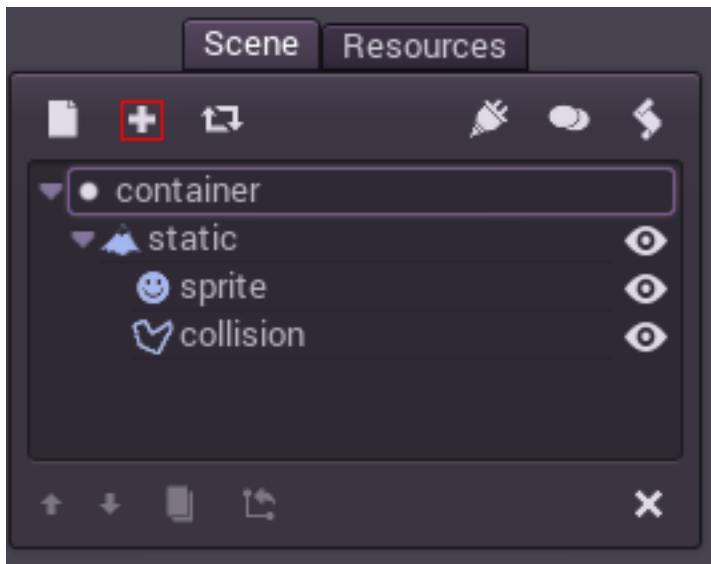
p=. [image4!](#)

[image5!](#)

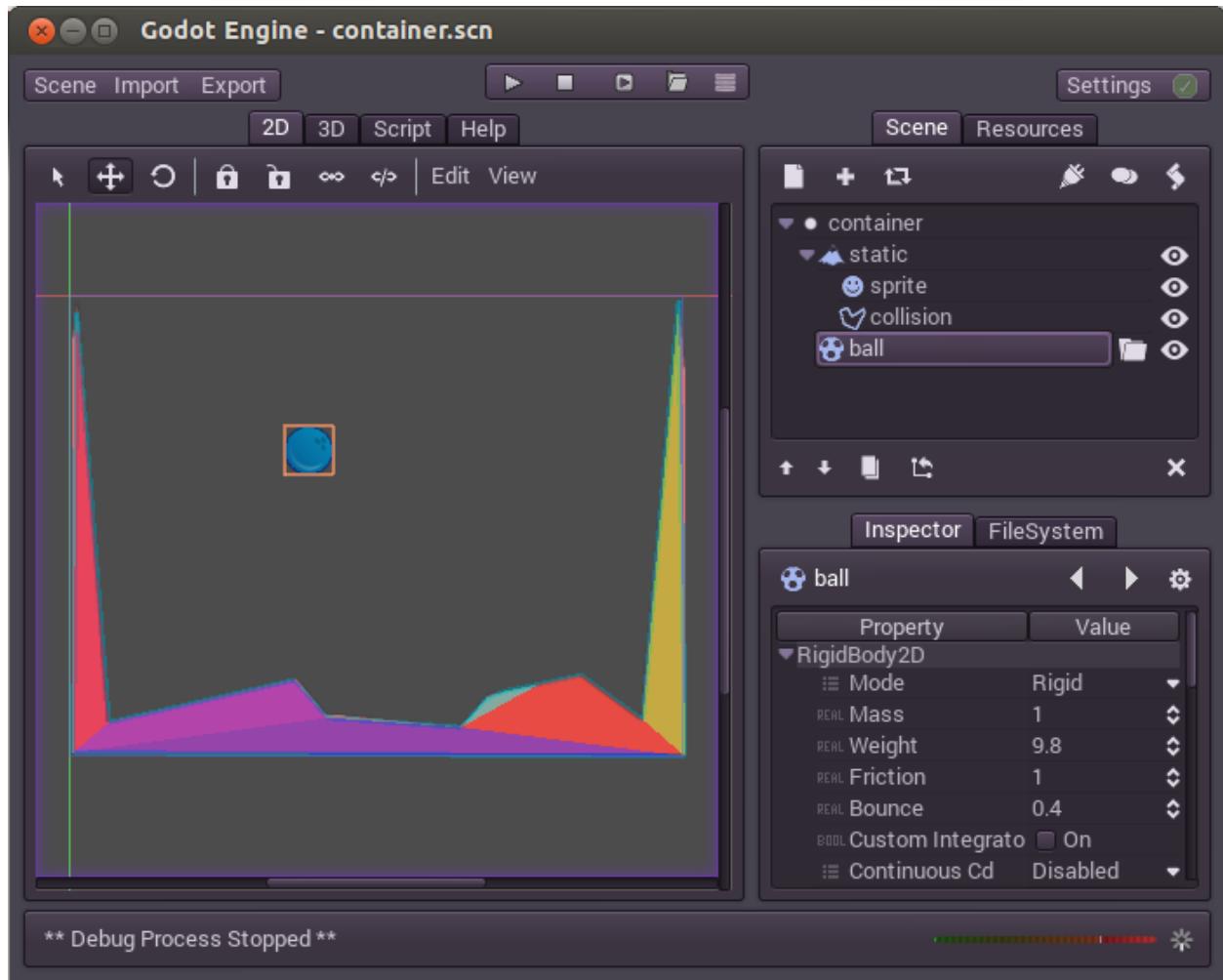
Open the container scene, then select the root node:



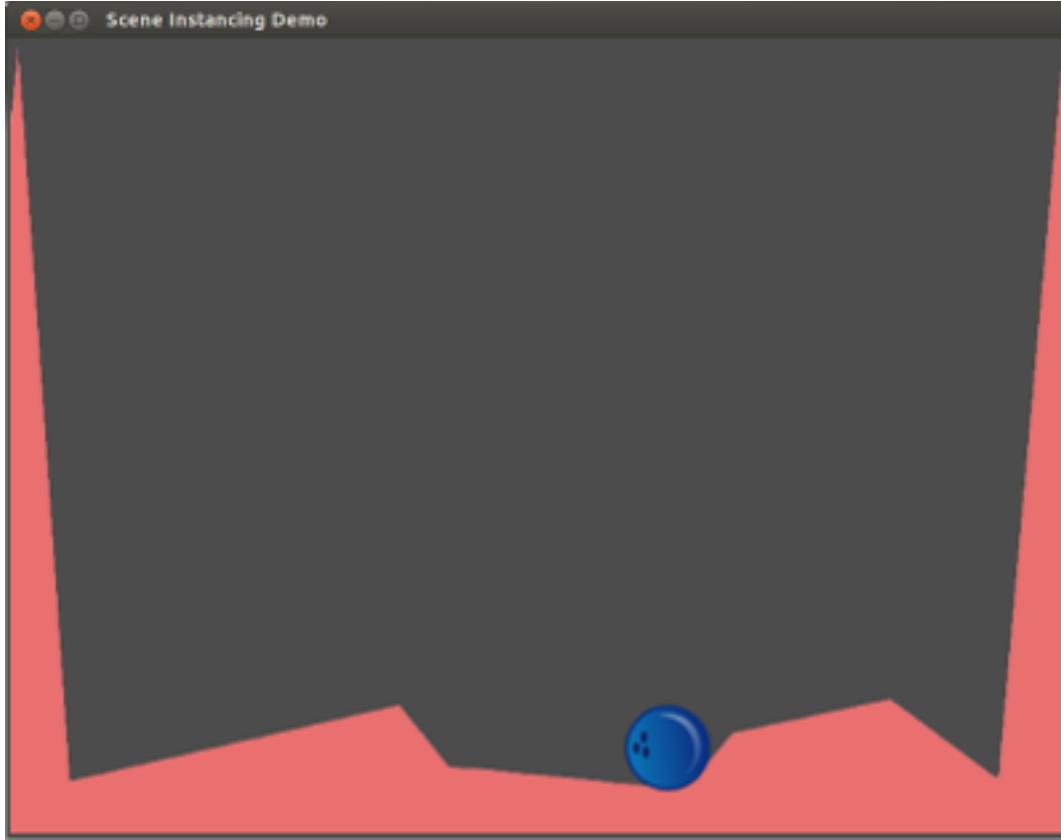
Afterwards, push the ‘+’ shaped button, this is the instancing button!



Select the ball scene (ball.scn), the ball should appear in the origin (0,0), move it to around the center of the scene, like this:



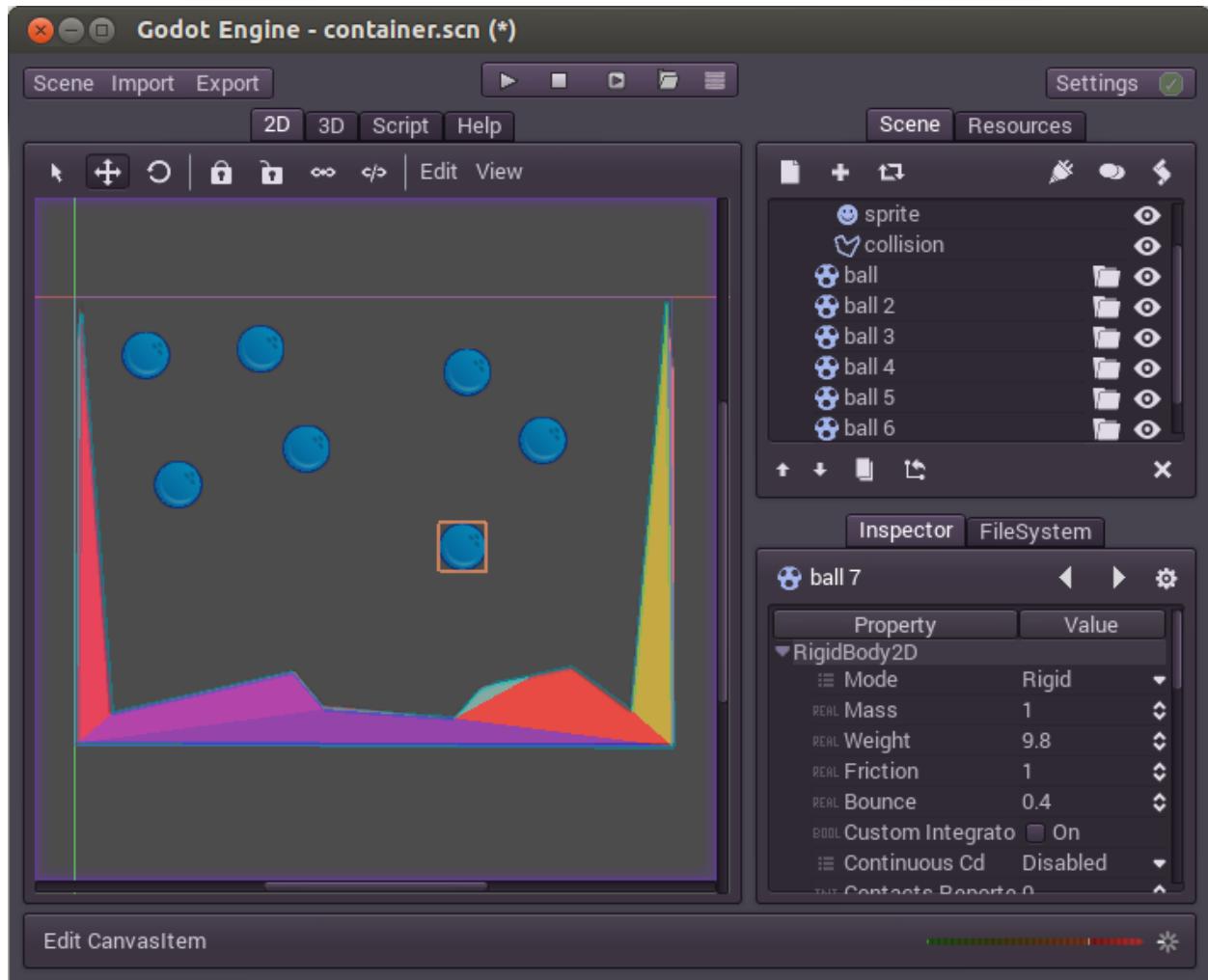
Press Play and Voila!



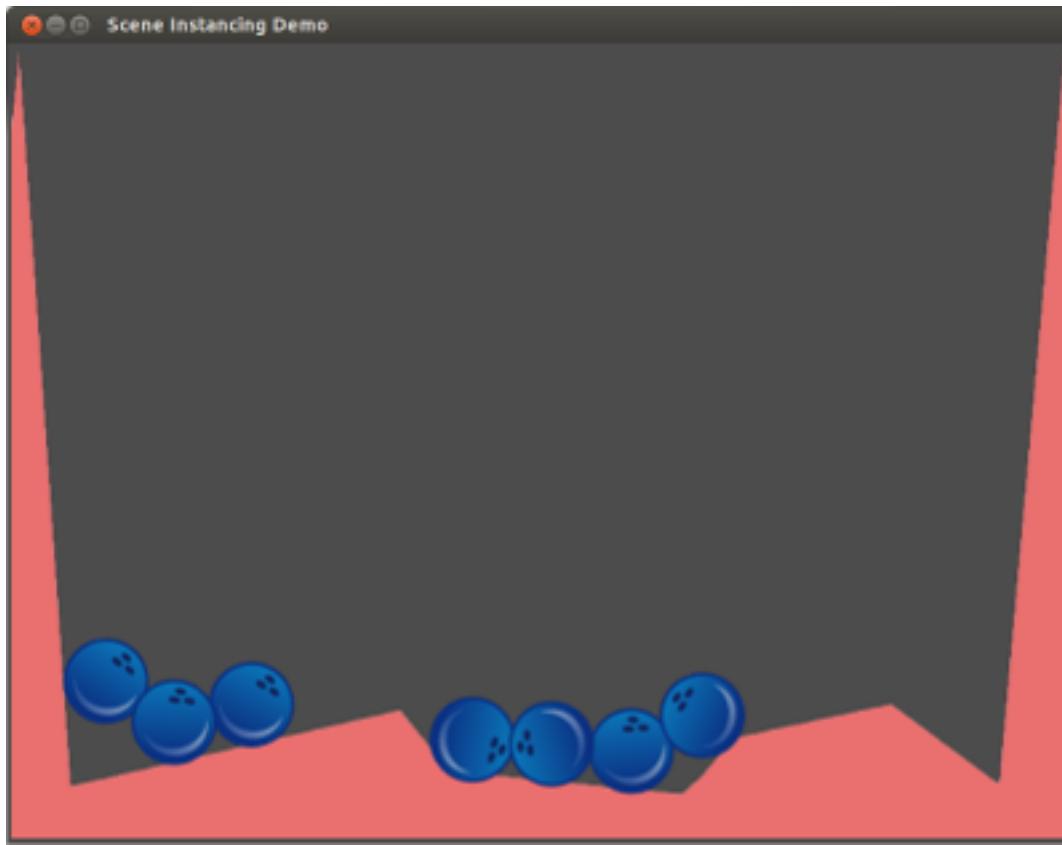
The instanced ball fell to the bottom of the pit.

### 1.2.3 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.

### 1.2.4 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:



The next it will happen is that a green “revert” button appears. When this button is present, it means we modified a property from the instanced scene to override for 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.

### 1.2.5 Conclusion

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

## 1.3 Instancing (continued)

### 1.3.1 Recap

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

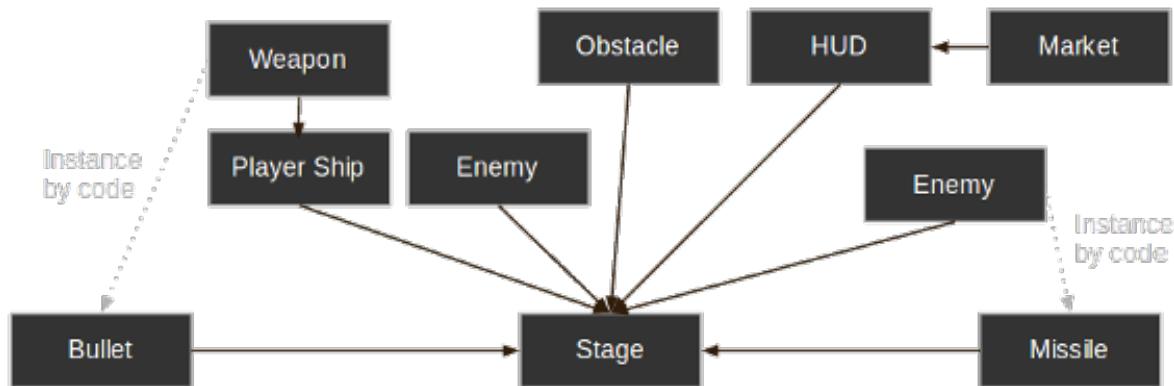
- The ability to subdivide scenes and make them easier to manage.
- A more flexible alternative to prefabs (and much more powerful given instances work at many levels).
- A way to design more complex game flows or even UIs (UI Elements are nodes in Godot too).

### 1.3.2 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. All the engine was designed from the ground 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 games in a more natural way. Start by imagining the visible elements in a game, the ones that can be named not by just a programmer but by anyone.

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

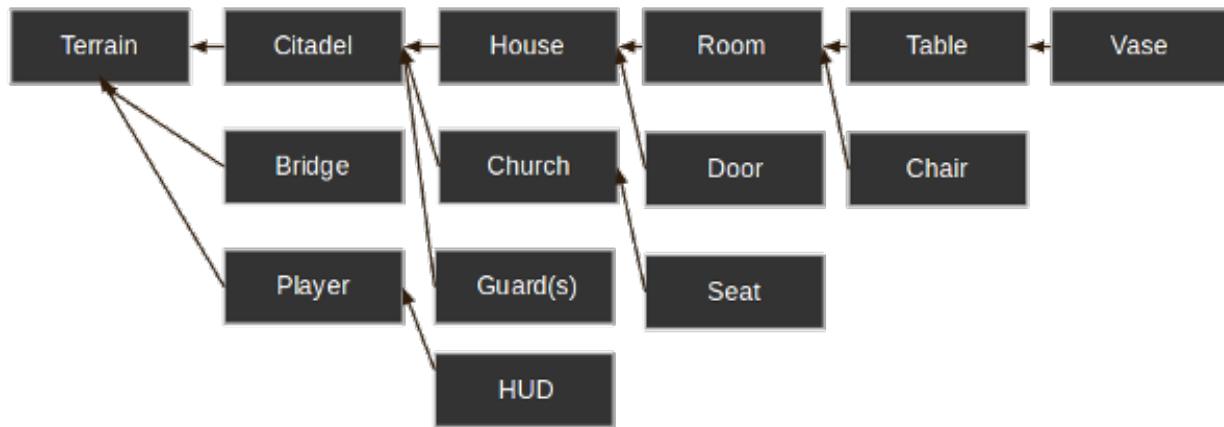


It's pretty easy to come up with a diagram like this for almost any kind of game. Just write down the elements that come to mind, and then the arrows that represent ownership.

Once this diagram exists, making a game is about creating a scene for each of those nodes, and use instancing (either by code [STRIKEOUT:more of that later] 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 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 are 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 only more scenes have to be made and instanced. The editor UI is also designed to be operated by non programmers too, so an usual team development process involves 3D or 2D artists, level designers, game designers, animators, etc all working with the editor interface.

### 1.3.3 Information Overload!

Do not worry to much, the important part of this tutorial is to create awareness on 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!

## 1.4 Scripting

### 1.4.1 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 in *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 extends from all the available engine classes.

## 1.4.2 GDScript

[[GDScript]] (click link for reference) 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 though, 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), making the resulting performance more than enough 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 for replacing a GDScript class with a C++ class without altering the rest of the game.

## 1.4.3 Scripting a Scene

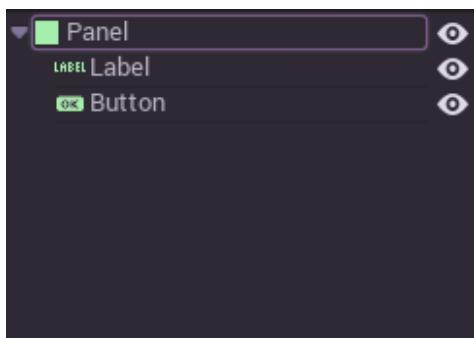
Before continuing, please make sure to read the [[GDScript]] reference. It's a simple language and the reference is short, should not take more than a few minutes to glance.

### 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:



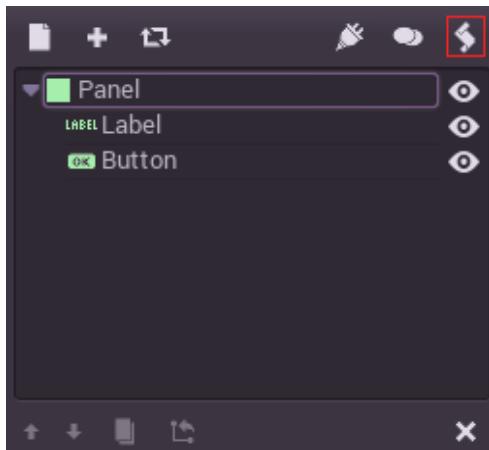
And try to make it look like this in the 2D editor, so it makes sense:



Finally, save the scene, a fitting name could be “sayhello.scn”

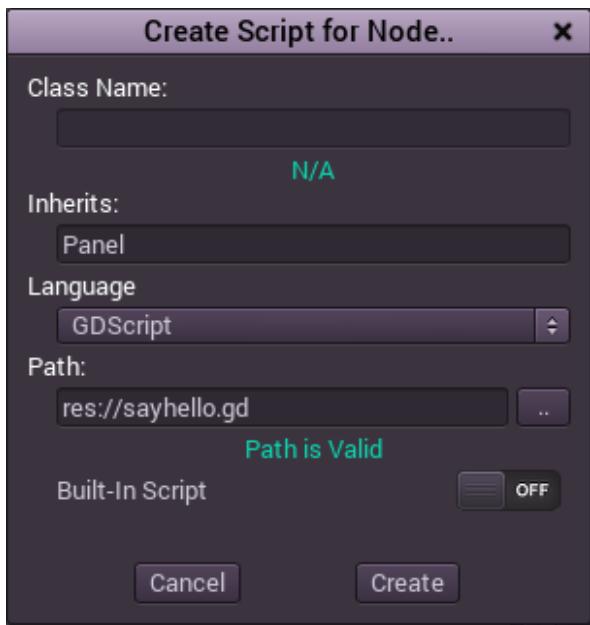
### Adding a Script

Select the Panel node, then press the “Add Script” Icon as follows:

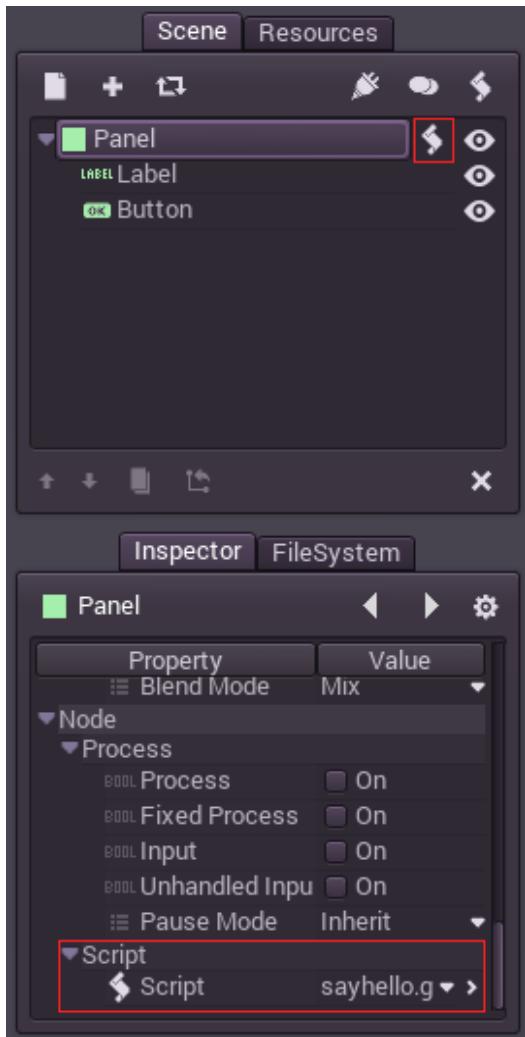


The script creation dialog will popup. This dialog allows to select the language, class name, etc. GDScript does not use class names in script files, so that field is not editable. The script should inherit from “Panel” (as it is meant to extend the node, which is of Panel type, this is automatically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

Select the filename for the script (if you saved the scene previously, one will be automatically generated as sayhello.gd) and push “Create”:



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, pushing the icon above should do it (although, the UI will take you directly to the Script editor screen). So, here's the template script:

The screenshot shows the Godot Script Editor interface. At the top, there is a menu bar with 'File', 'Edit', 'Search', 'Debug', and 'Window'. Below the menu is a toolbar with buttons for '2D', '3D', 'Script', and 'Help'. The main area is titled 'sayhello.gd' and contains the following GDScript code:

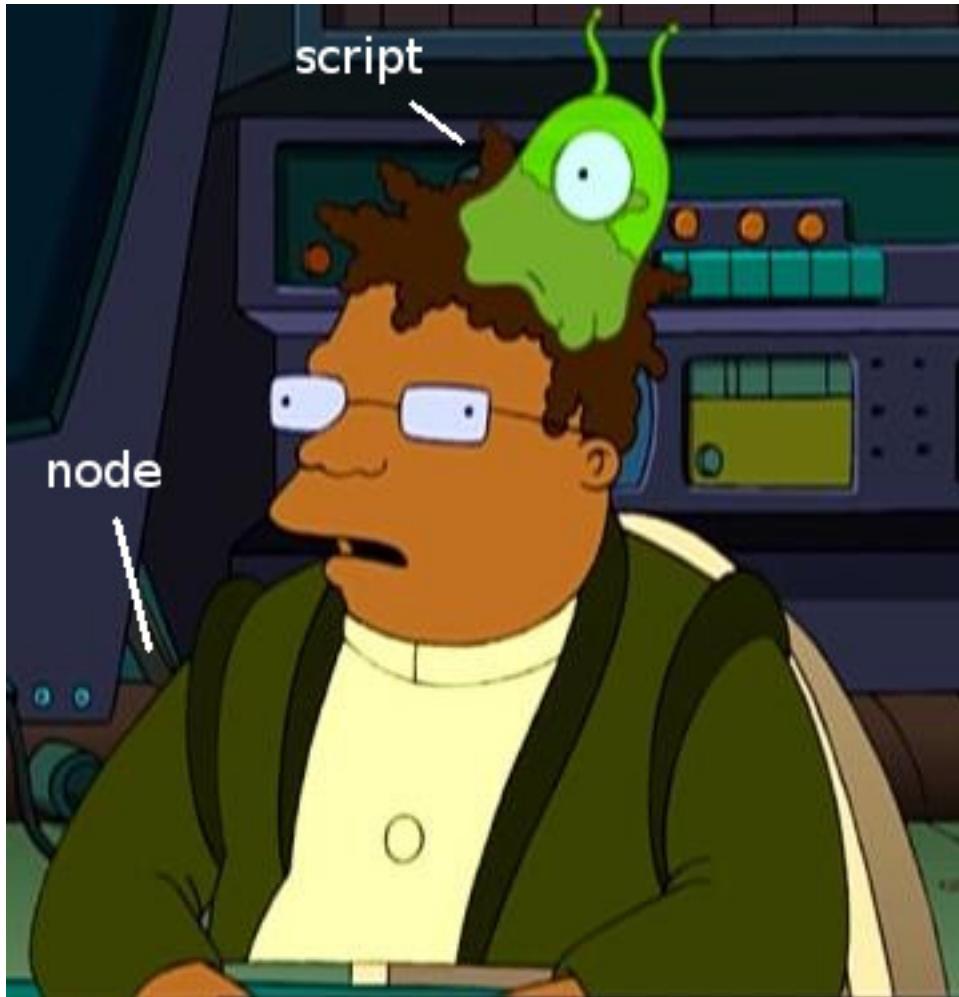
```
01 |
02 extends Panel
03
04 # member variables here, example:
05 # var a=2
06 # var b="textvar"
07
08 func _ready():
09 »   # Initialization here
10 »   pass
11
12
13
```

At the bottom right of the editor window, it says 'Line: 1, Col: 0'.

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

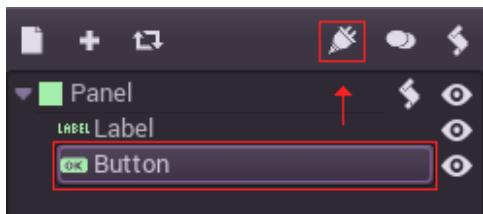
A script basically adds a behavior to a node. It is used to control the node functions as well as other nodes (children, parent, siblings, etc). The local scope of the script is the node (just like in regular inheritance) and the virtual functions of the node are captured by the script.



## 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.

There is a GUI for connecting signals, just select the node and press the “Signals” button:



Which will show the list of signals a Button can emit.



But this example will not use it. We don't want to make things *too* easy. So please close that screen!

In any case, at this point it is clear that that we are interested in the “pressed” signal, so instead of doing it with the visual interface, the connection will be done using code.

For this, there is a function that is probably the one that Godot programmers will use the most, this is `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:

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

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

```
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 `connect`.

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

The final script should look like this:

```
extends Panel

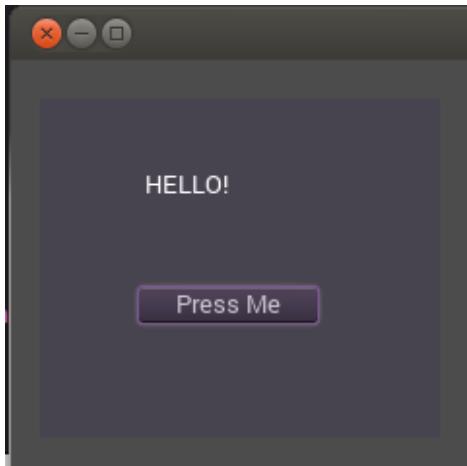
# member variables here, example:
```

```
# 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 mistake in this tutorial, let's clarify again that `get_node(path)` works by returning the immediate children to 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* was a child of *Label*, the code to obtain it would be:

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

And, also, try to remember that nodes are referenced by name, not by type.

## 1.5 Scripting (continued)

### 1.5.1 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 time 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:

```
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:

```
extends Label

var accum=0

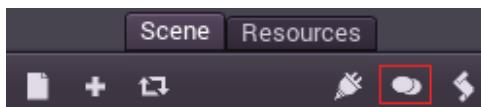
func _ready():
    set_process(true)

func _process(delta):
    accum+=delta
    set_text(str(accum))
```

Which will show a counter increasing each second.

## 1.5.2 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:



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

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

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

```
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()`:

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

More will be added about `SceneTree` later.

## 1.5.3 Notifications

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

```
func _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()...")
```

The documentation of each class in the [class list](#) shows the notifications it can receive. However, again, for most cases script provides simpler overrideable functions.

## 1.5.4 Overrideable Functions

As mentioned before, it's better to use these functions. Nodes provide many useful overrideable functions, which are described as follows:

```
func _enter_tree():
    pass # When the node enters the _Scene Tree_, it becomes active and this function is called. Children nodes have all entered their tree.

func _ready():
    pass # This function is called after _enter_tree_, but it ensures that all children nodes have also entered their tree.

func _exit_tree():
    pass # When the node exists the _Scene Tree_, this function is called. Children nodes have all exited their tree.

func _process(delta):
    pass # When set_process() is enabled, this is called every frame

func _fixed_process(delta):
    pass # When set_fixed_process() is enabled, this is called every physics frame

func _paused():
    pass # Called when game is paused, after this call, the node will not receive any more process calls

func _unpaused():
    pass # Called when game is unpaused
```

## 1.5.5 Creating Nodes

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

```
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:

```
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 might want to delete a node that is currently “blocked” this means, the node 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 `queue_free()` instead. This erases the node during idle, safely.

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

## 1.5.6 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.

```
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.

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

But ‘scene’ is still not 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:

```
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 specially useful, for example, to instance several enemies, bullets, etc. quickly in the active scene.

## 1.6 Simple 2D game (Pong!)

### 1.6.1 Pong

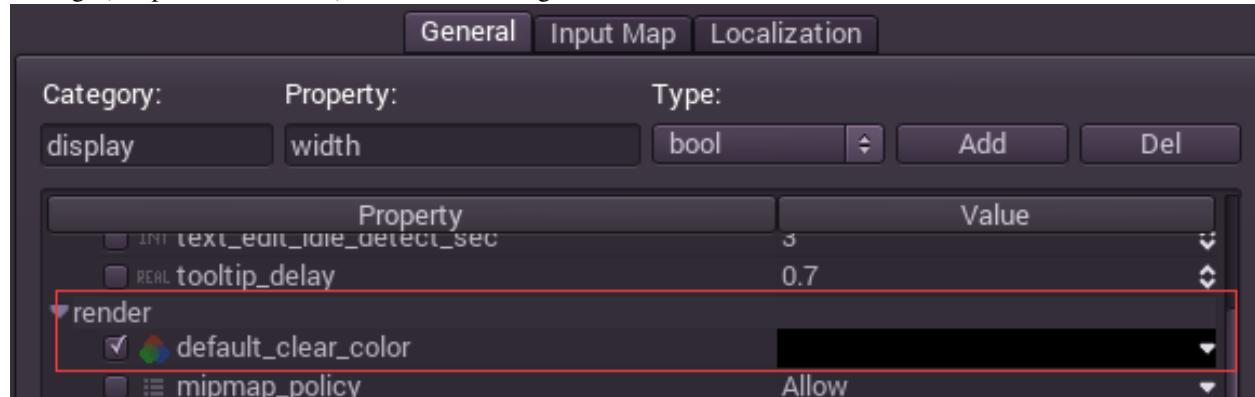
In this simple tutorial, a basic game of Pong will be created. There are plenty of more complex examples in the demos included with the engine, but this should get introduced to basic functionality for 2D Games.

### 1.6.2 Assets

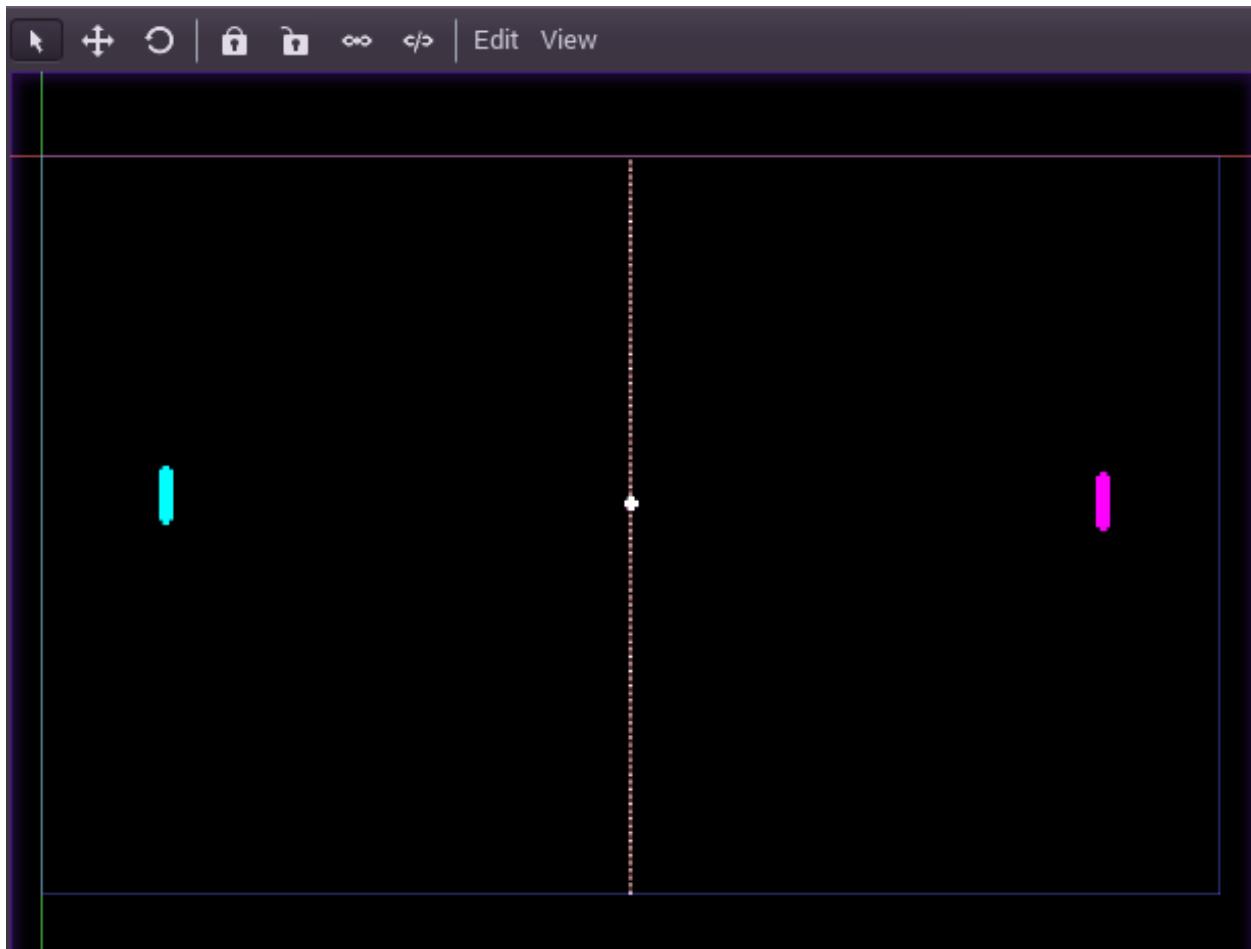
Some assets are included for this tutorial, the attachment:pong\_assets.zip.

### 1.6.3 Scene Setup

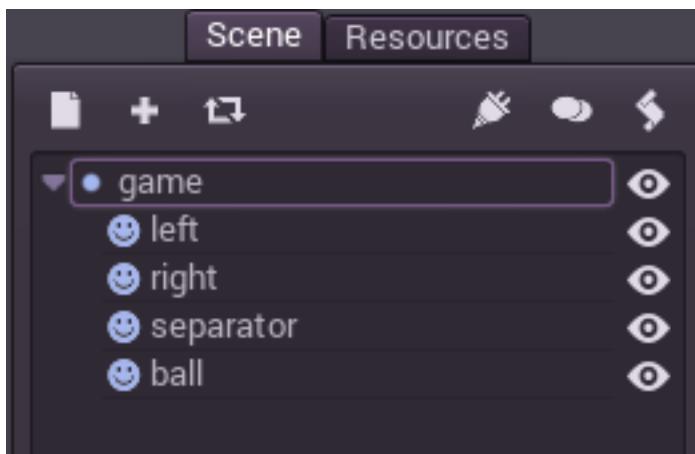
For the sake of the old times, the game will be in 640x400 pixels resolution. This can be configured in the Project Settings (see previous tutorials). The default background color should be set to black:



Create a [[class\_node2d]] node for the project root. Node2D is the base type for the 2D engine. After this, add some sprites ([[class\_sprite]]) node) and set each to the corresponding texture. The final scene layout should look similar to this (note: the ball is in the middle!):



The scene tree should, then look similar to this:



Save the scene as “pong.scn” and set it as the main scene in the project properties.

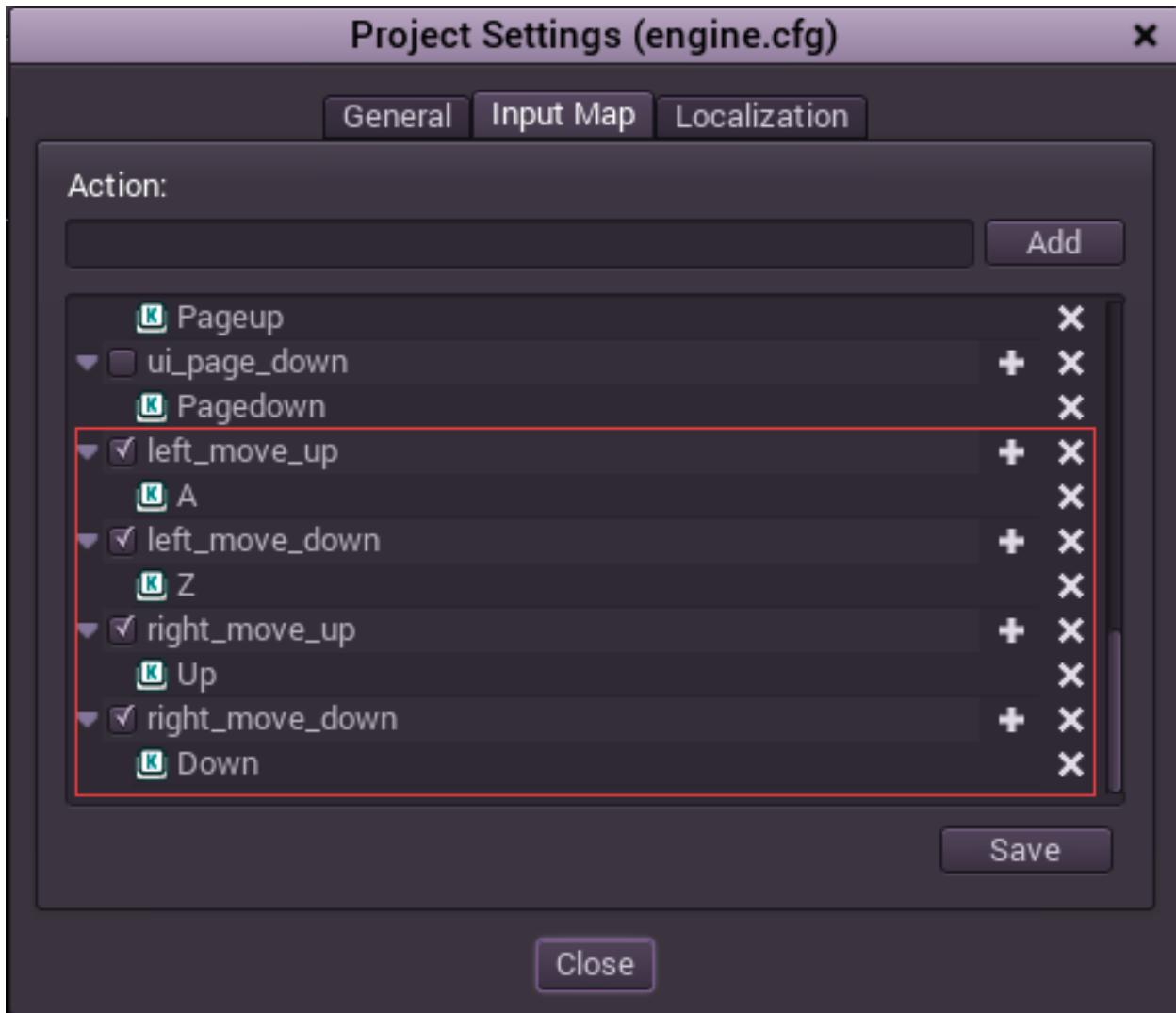
#### 1.6.4 Input Actions Setup

There are so many input methods for video games... Keyboard, Joypad, Mouse, Touchscreen (Multitouch). Yet this is pong. The only input that matters is for the pads going up and down.

Handling all possible input methods can be very frustrating and take a lot of code. The fact that most games allow controller customization makes this worse. For this, Godot created the “Input Actions”. An action is defined, then input methods that trigger it are added.

Open the project properties dialog again, but this time move to the “Input Map” tab.

On it, add 4 actions: “left\_move\_up”, “left\_move\_down”, “right\_move\_up”, “right\_move\_down”. Assign the keys that you desire. A/Z for left and Up/Down as keys should work in most cases.



## 1.6.5 Script

Create a script for the root node of the scene and open it (should have been explained in the previous tutorial!). The script will inherit Node2D:

```
extends Node2D

func _ready():
    pass
```

In the constructor, two things will be done. The first is to enable processing, and the second to store some useful values. Such values are the dimensions of the screen and the pad:

```
extends Node2D

var screen_size
var pad_size

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

Then, some variables used for in-game will be added:

```
#speed of the ball (in pixels/second)

var ball_speed = 80
#direction of the ball (normal vector)

var direction = Vector2(-1,0)
#constant for pad speed (also in pixels/second)

const PAD_SPEED = 150
```

Finally, the process function:

```
func _process(delta):
```

Get some useful values for computation. The first is the ball position (from the node), the second is the rectangles (Rect2) of the pads. Sprites by default center the textures, so a small adjustment of size/2 must be added.

```
var ball_pos = get_node("ball").get_pos()
var left_rect = Rect2( get_node("left").get_pos() - pad_size/2, pad_size )
var right_rect = Rect2( get_node("right").get_pos() - pad_size/2, pad_size )
```

Since the ball pos was obtained, integrating it should be simple:

```
ball_pos+=direction*ball_speed*delta
```

Then, now that the ball has a new position, it should be tested against everything. First, the floor and the roof:

```
if ( (ball_pos.y<0 and direction.y <0) or (ball_pos.y>screen_size.y and direction.y>0) ):
    direction.y = -direction.y
```

If one of the pads was touched, change direction and increase speed a little.

```
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
    ball_speed*=1.1
    direction.y=randf()*2.0-1
    direction = direction.normalized()
```

If the ball went out of the screen, it's game over. Game restarts:

```
if (ball_pos.x<0 or ball_pos.x>screen_size.x):
    ball_pos=screen_size*0.5 #ball goes to screen center
    ball_speed=80
    direction=Vector2(-1,0)
```

Once everything was done with the ball, the node is updated with the new position:

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

Only updating the pads according to player input. the Input class is really useful here:

```
#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)
```

And that's it! a simple Pong was written with a few lines of code.

## 1.7 GUI tutorial

### 1.7.1 Introduction

If there is something that most programmers hate with passion, that 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 (so 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 in 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.

### 1.7.2 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 interface quickly and visually.

### 1.7.3 Input and Drawing

Controls receive input events by means of the `_input_event()` callback. Only one control, the one in focus, will receive keyboard/joypad events (see `set_focus_mode()` and `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 `_draw()` callback will be received at the beginning and every time the control needs to be redrawn (programmer needs to call `update()` to enqueue the `CanvasItem` for redraw). If the control is not visible (yet another `CanvasItem` property), the control does not receive any input.

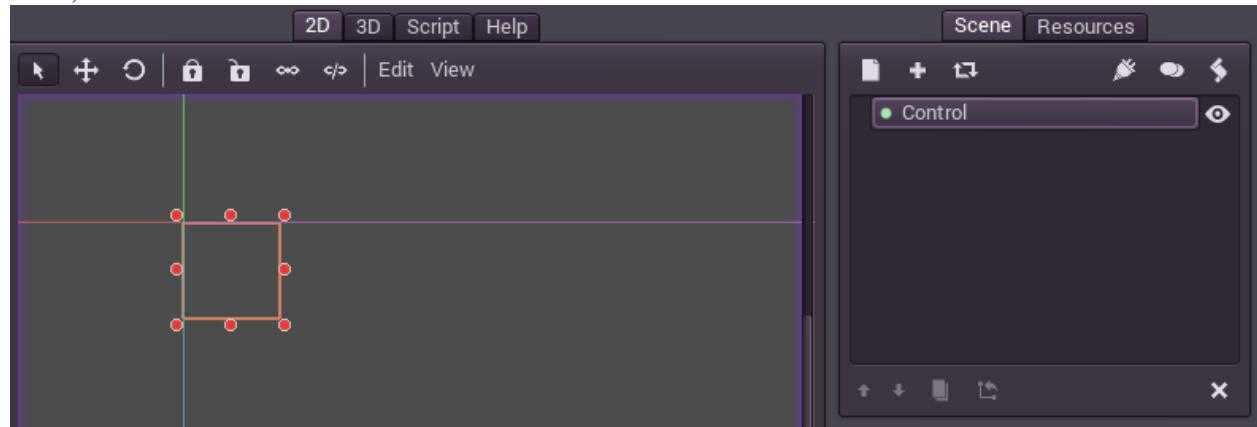
In general though, the programmer does not need to deal with drawing and input events directly when building UIs, (that is more useful when creating custom controls). Instead, controls emit different kinds of signals with contextual information for when action occurs. For example, a `Button` emits a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

### 1.7.4 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 work, 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 just 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:

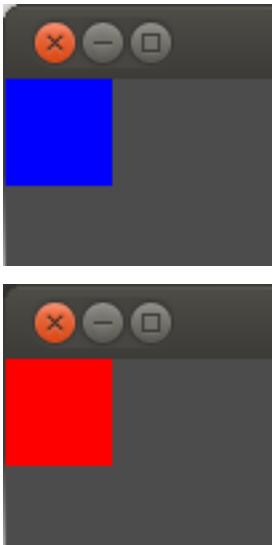
```
extends Control

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/taped, it will go from blue to red. That synergy between the events and drawing is pretty much how most controls work internally.



### 1.7.5 UI Complexity

As mentioned before, Godot includes dozens of controls ready for using in a user interface. Such controls are divided in 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.

### 1.7.6 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 texture buttons, 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 [[GUI Repositioning]] tutorial for more info).

In any case, it will happen often that even for simple games, more complex UI behaviors will be 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 anyway).

## 1.7.7 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 [[GUI Repositioning]] tutorial for more info).

## 1.8 Splash screen

### 1.8.1 Tutorial

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

Following is a file with the assets that will be used:

attachment:robisplash\_assets.zip

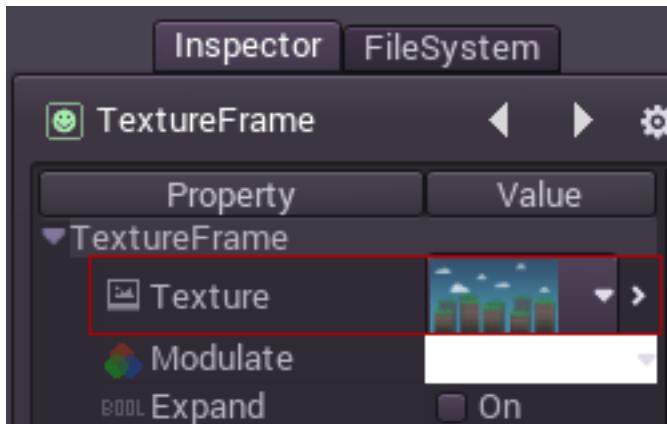
### 1.8.2 Setting Up

Create a scene with screen resolution 800x450, and set it up like this:

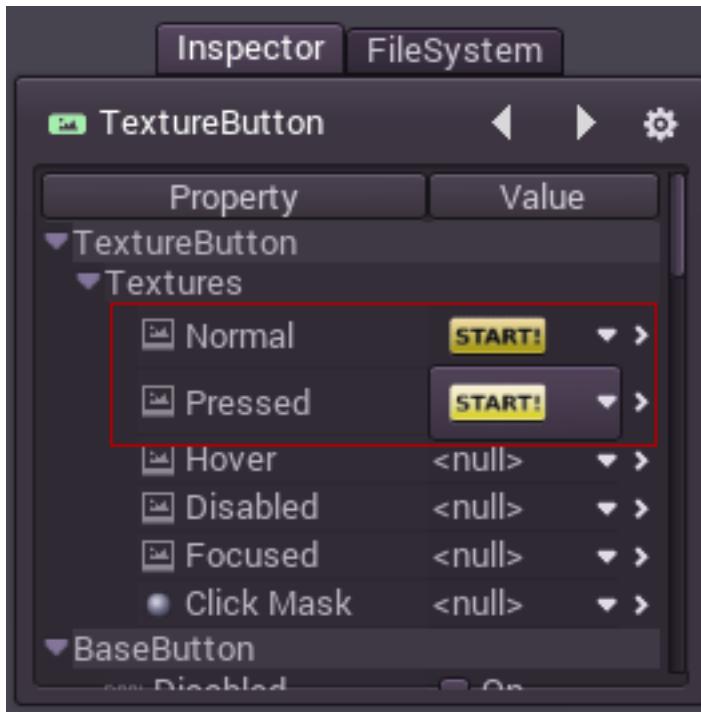




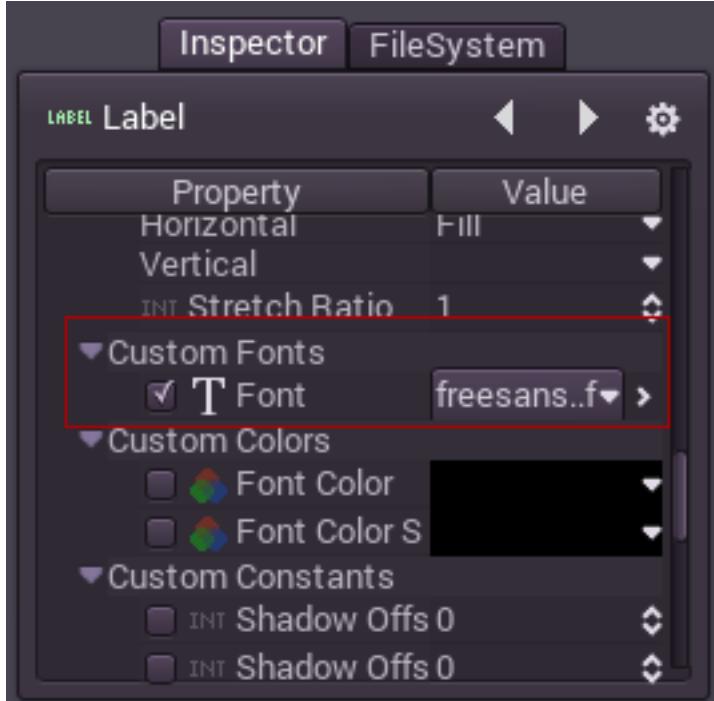
The nodes ‘background’ and “logo” are of [TextureFrame](#) type. These have a special property for setting the texture to be displayed, just load the corresponding file.



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](#). Labels can be set a custom font by editing the following property:



As a side note, the font was imported from a TTF, there is a [[Importing Fonts]] for importing fonts.

## 1.9 Animations

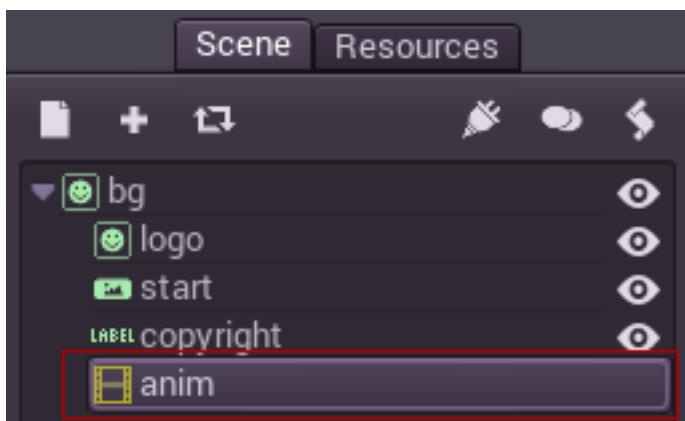
### 1.9.1 Introduction

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

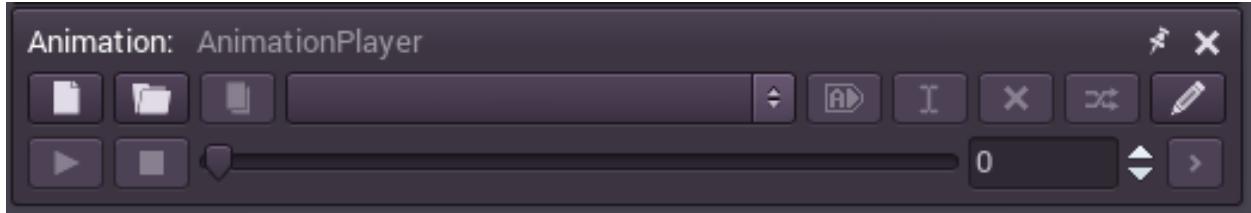
To begin, let's just use the scene from the previous tutorial (splash screen). The goal will be to add a simple animation to it. Here's a copy just in case: [attachment:robisplash.zip](#)

### 1.9.2 Creating the Animation

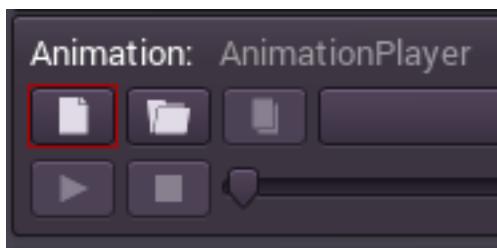
First of all, add an `AnimationPlayer` node to the scene, make it a child of `bg` (the root node):



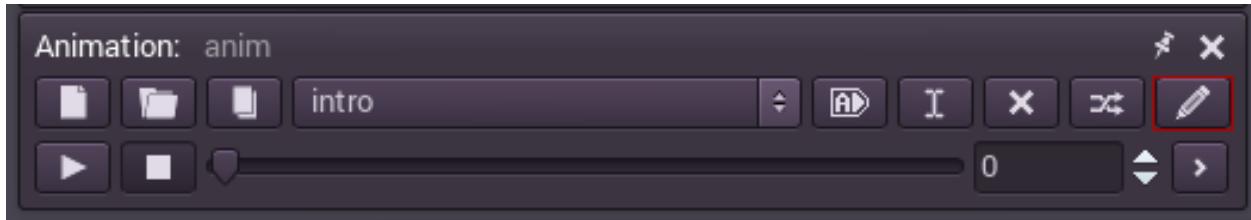
When a node of this type is selected, the animation editor panel will appear:



So, it's time to create a new animation! Press the new animation button and name the animation "intro".

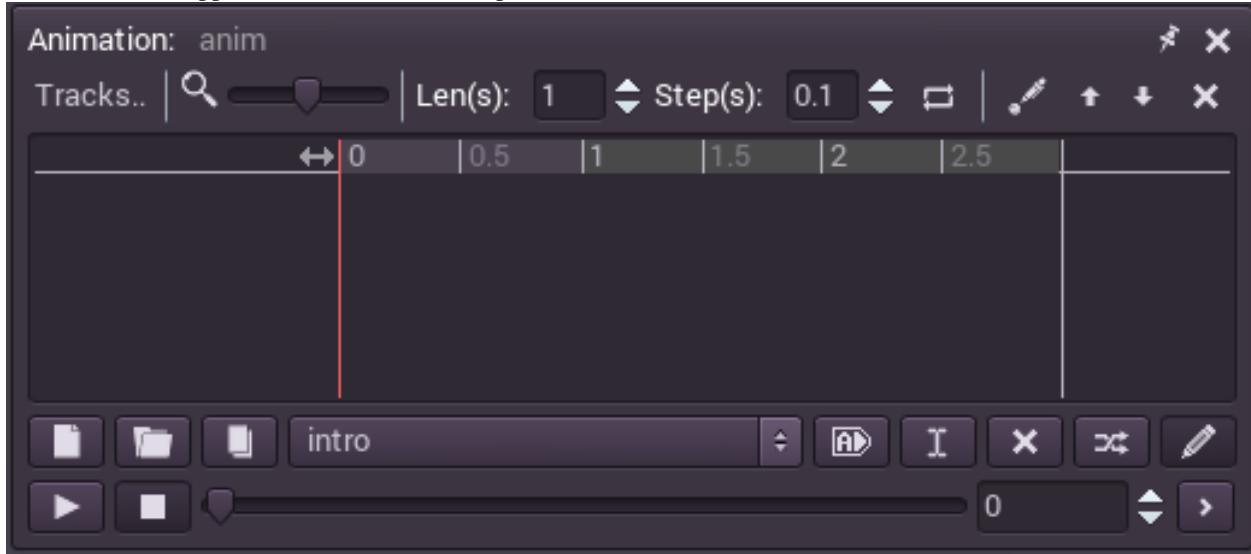


After the animation has been created, then it's time to edit it, by pressing the "edit" button:



### 1.9.3 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.



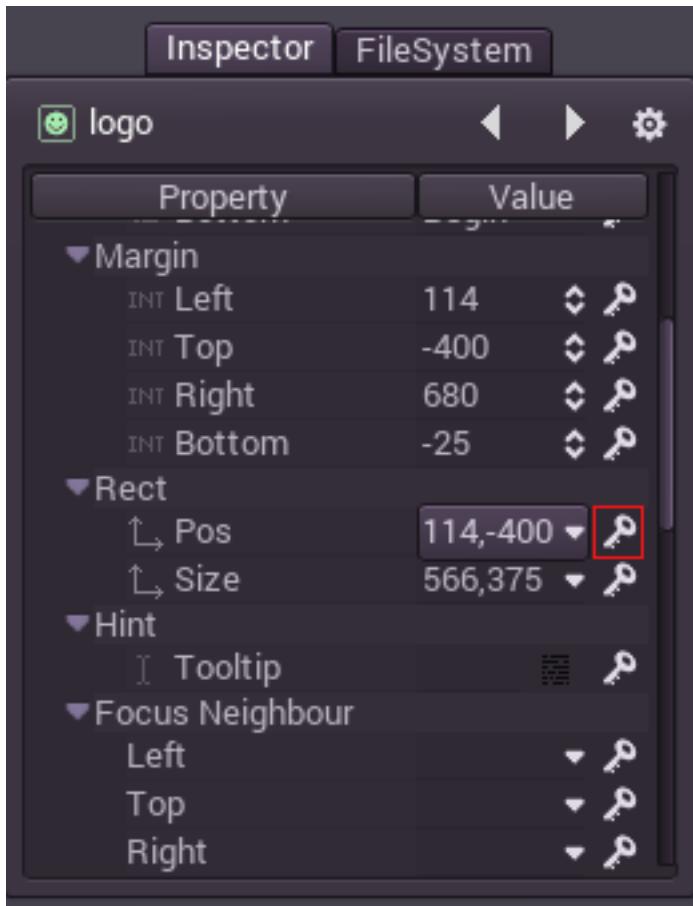
But the second, and most important, is that the property editor enters into “animation editing” mode. In this mode, a key icon appears next to every property of the property editor. This means that, in Godot, *any property of any object* can be animated:



#### 1.9.4 Making the Logo Appear

Next, 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.

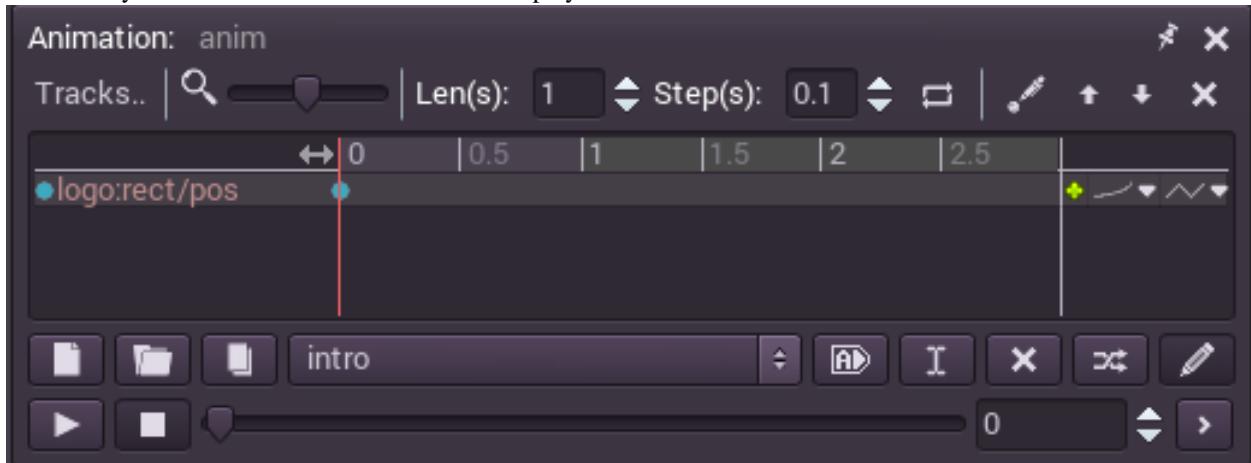
Once in this position, press the key button next to the property:



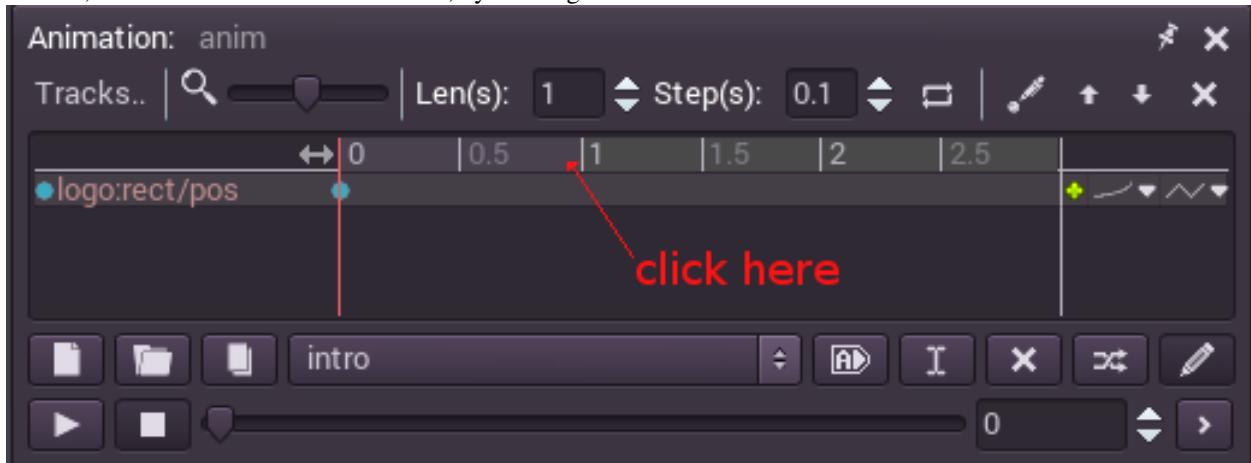
As the track is new, a dialog will appear asking to create it. Confirm it!



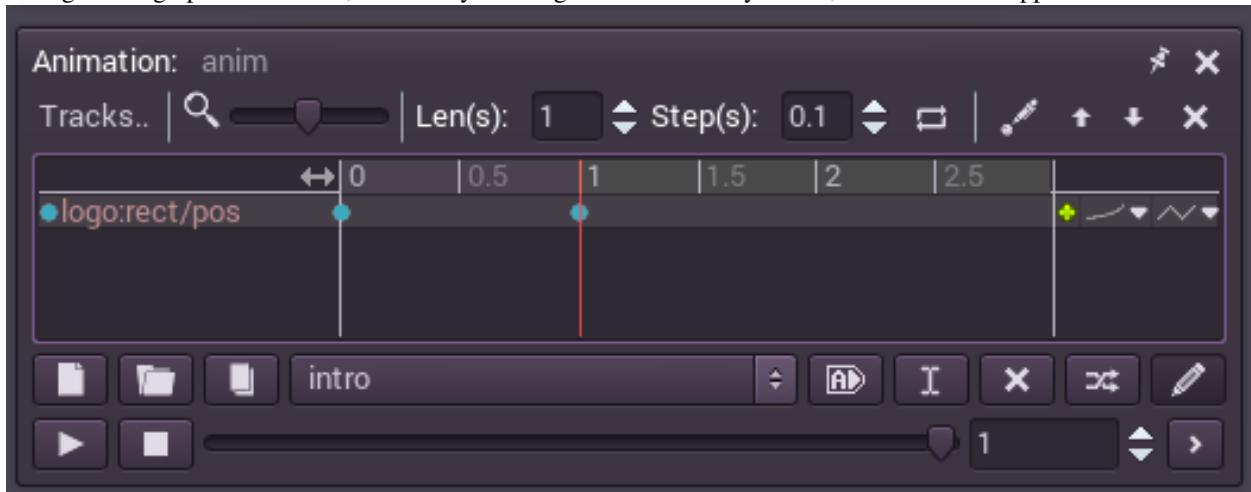
And the keyframe will be added in the animation player editor:



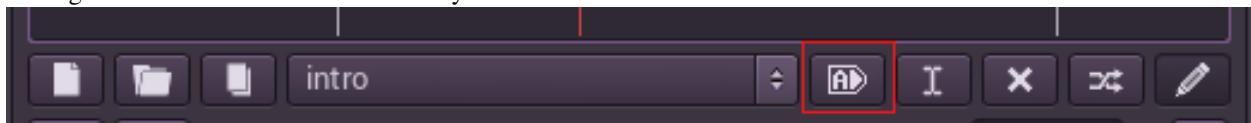
Second, move the editor cursor to the end, by clicking here:



Change the logo position to 114,0 and 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:

## 1.10 Resources

### 1.10.1 Nodes and Resources

So far, [Node](#) have been the most important datatype in Godot, as most of the behaviors and features of the engine are implemented through them. There is, though, another datatype that is equally as important. That is [Resource](#).

Where *Nodes* focus on behaviors, such as drawing a sprite, drawing a 3D model, physics, GUI controls, etc,

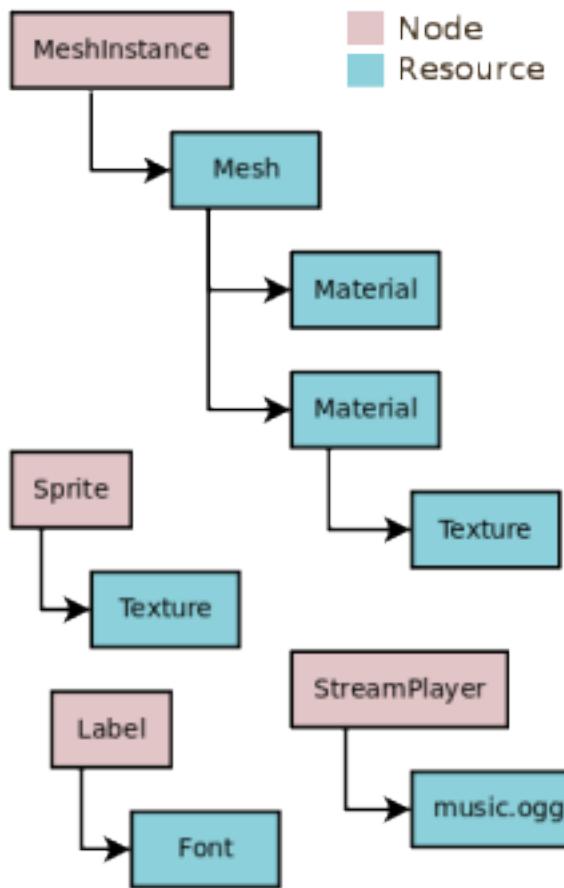
**Resources** are mere **data containers**. This means that they don't do any action nor process any information. Resources just contain data.

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.

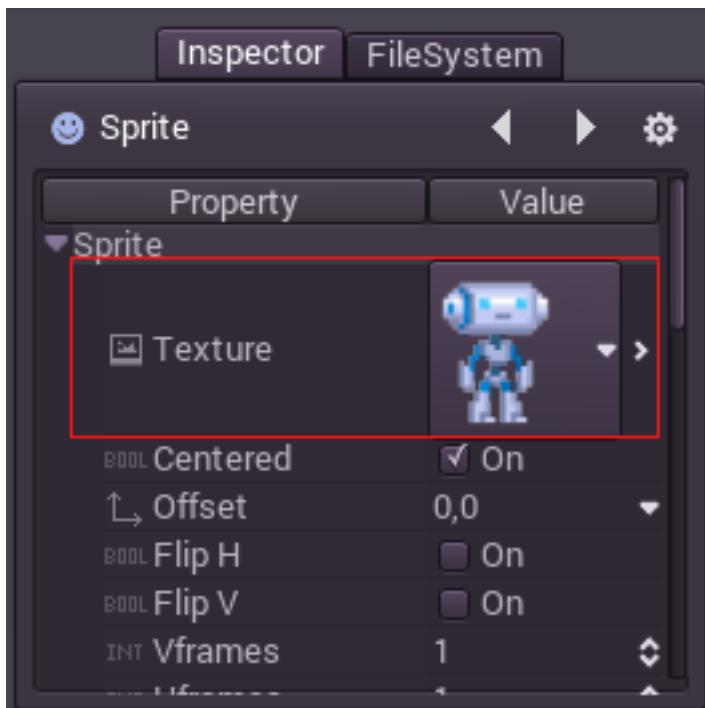
Typically, every object in Godot (Node, Resource, or anything else) can export properties, properties can be of many types (like a string, integer, Vector2, etc) and one of those types can be a resource. This means that both nodes and resources can contain resources as properties. To make it a little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



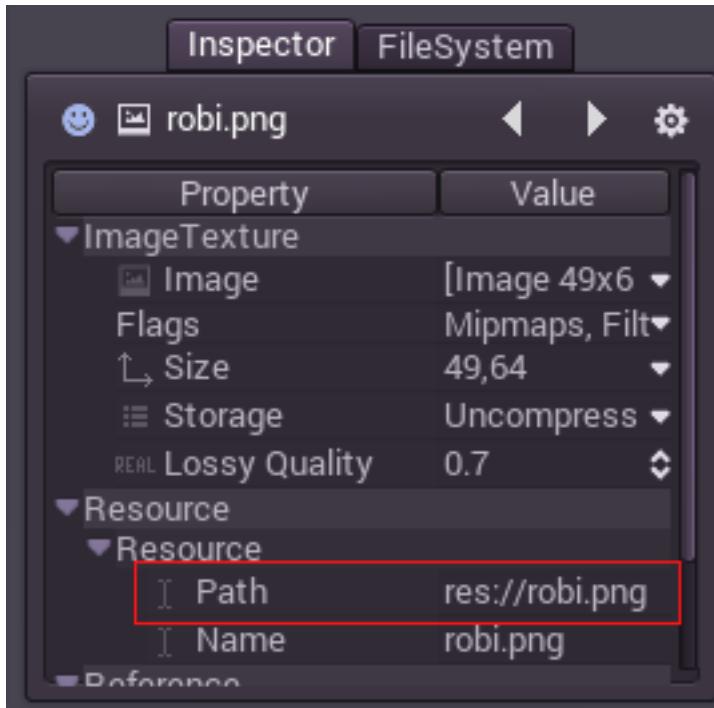
### 1.10.2 External vs Built-In

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 the right side of the preview, allows to view and edit the resources properties. One of the properties (path) shows where did it come from. In this case, it came from a png image.



When the resource comes from a file, it is considered an *external* resource. If the path property is erased (or never had a path to begin with), it is then 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 still always be

loaded once. That means, different Robi robot scenes instanced at the same time will still share the same image.

### 1.10.3 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:

```
func _ready():
    var res = load("res://robi.png") h1. 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.

```
func _ready():
    var res = preload("res://robi.png") h1. resource is loaded at compile time
    get_node("sprite").set_texture(res)
```

### 1.10.4 Loading Scenes

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

To obtain an instance of the scene, the method `instance()` must be used.

```
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 `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.

### 1.10.5 Freeing Resources

Resource extends from [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.

### 1.10.6 Scripting

Like any object in Godot, not just nodes, Resources can be scripted too. However, there isn't generally much of a win, as resources are just data containers.

## 1.11 Filesystem

### 1.11.1 Introduction

Filesystem usage is yet another hot topic in engine development. This means, where are assets stored, how are they accessed, how do multiple programmers edit the same repository, etc.

Initial versions of the engine (and previous iterations before it was named Godot) used a database. Assets were stored there and assigned an ID. Other approaches were tested, too, with local databases, files with metadata, etc. To say truth, and after a long time, simplicity proved to be best and Godot stores all assets as files in the filesystem.

### 1.11.2 Implementation

Godot stores resources to 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 path to that resource is included. If it 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 saved with the character textures bundled inside.

Metadata files were also dropped and the whole engine design tries to avoid them. The reason for this is simple, existing asset managers and VCSs are just much better than anything we can implement, so Godot tries the best to play along with SVN, Git, Mercurial, Perforce, etc.

### 1.11.3 engine.cfg

The mere existence of this file marks that there is a Godot project in that directory and all sub-directories. This file contains the project configuration in plain text, win.ini style, though it will work to mark the existence of a project even if the file is empty.

Example of a filesystem:

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

### 1.11.4 Directory Delimiter

Godot only supports “/” as a directory 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.

### 1.11.5 Resource Path

For accessing resources, using the host OS filesystem 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 filesystem 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 filesystem will become read-only and writing will no longer be permitted.

### 1.11.6 User Path

Writing to disk is still needed often, from doing a savegame to downloading content packs. For this, the engine ensures that there is a special path ““user://”“ that is always writable.

### 1.11.7 Host Filesystem

Of course, opening the host filesystem always works, as this is always useful when Godot is used to write tools, but for shipped projects this is discouraged and may not even be supported in some platforms.

### 1.11.8 Drawbacks

Not everything is rosy. Using resources and files and the plain filesystem has two main drawbacks. The first is that moving assets around (renaming them or moving them from a directory to another inside the project) once they are referenced is not that easy. If this is done, then dependencies will need to be re-satisfied upon load.

The second is that under Windows or OSX, file access is case insensitive. If a developer works in this operating system and saves a file like “myfile.PNG”, then references it as “myfile.png”, it will work there, but not on any other platform, such as Linux, Android, etc. It may also not work on exported binaries, which use a compressed package for files.

Because of this, please instruct your team to use a specific naming convention for files when working with Godot!

## 1.12 SceneTree

### 1.12.1 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 deserves going a little more into depth. In fact, the scene system is not even a core component of Godot, as it is possible to skip it and make a script (or C++ code) that talks directly to the [[Servers]]. But making a game that way would be a lot of work and is reserved for other uses.

### 1.12.2 MainLoop

The way Godot works internally is as follows. There is the 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 (frame-synchronized 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.

### 1.12.3 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`, when a scene is first opened, it's added as a child of it 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()`.

### 1.12.4 Root Viewport

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

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

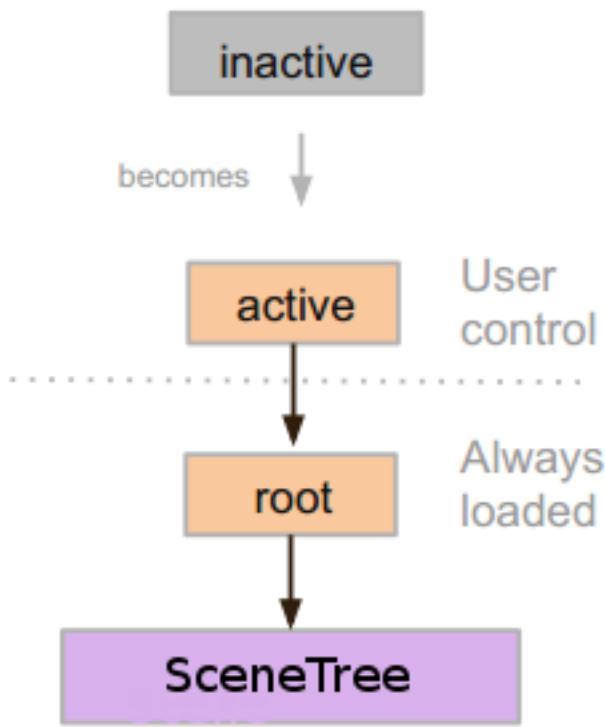
This node contains the main viewport, anything that is a child of a `Viewport` is drawn inside of it by default, so it makes sense that the top of all nodes is always a node of this type, otherwise nothing would be seen!

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`.

### 1.12.5 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, 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 it.

### 1.12.6 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 less order will get notified before the current node.



### 1.12.7 “Becoming Active” by entering the *Scene Tree* In Detail

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

### 1.12.8 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:

```
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]].

## 1.13 Singletons (AutoLoad)

### 1.13.1 Introduction

Scene Singletons are very useful things, as they represent a very common use case, but it's not clear at the beginning where their value is.

The scene system is very useful, but by itself it has a few drawbacks:

- There is no “common” place to store information (such as core, items obtained, etc) between two scenes.
- It is possible to make a scene that loads other scenes as children and frees them, while keeping that information, but then if that is done, it's not possible to run a scene alone by itself and expect it to work
- It is also possible to store persistent information to disk in ‘user://’ and have scenes always load it, but saving/loading that while changing scenes is cumbersome.

So, after using Godot for a while, it becomes clear that it is necessary to have parts of a scene that:

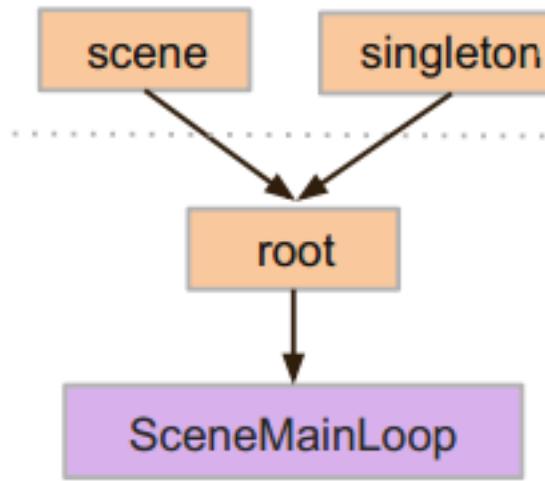
- Are always loaded, no matter which scene is opened from the editor.
- Can keep global variables, such as player information, items, money, etc.
- Can handle switching of scenes and transitions.
- Just have something that acts like a singleton, since GDScript does not support global variables by design.

For this, the option for auto-loading nodes and scripts exists.

### 1.13.2 Autoload

Autoload can be a scene, or a script that inherits from Node (a Node will be created and the script will be set to it). They are added to the project in the Scene [STRIKEOUT:> Project Settings]> AutoLoad tab.

Each autoload needs a name, this name will be the node name, and the node will be always added to the root viewport before any scene is loaded.



This means, that a for a singleton named “playervariables”, any node can access it by requesting:

```
var player_vars = get_node("/root/playervariables")
```

### 1.13.3 Custom Scene Switcher

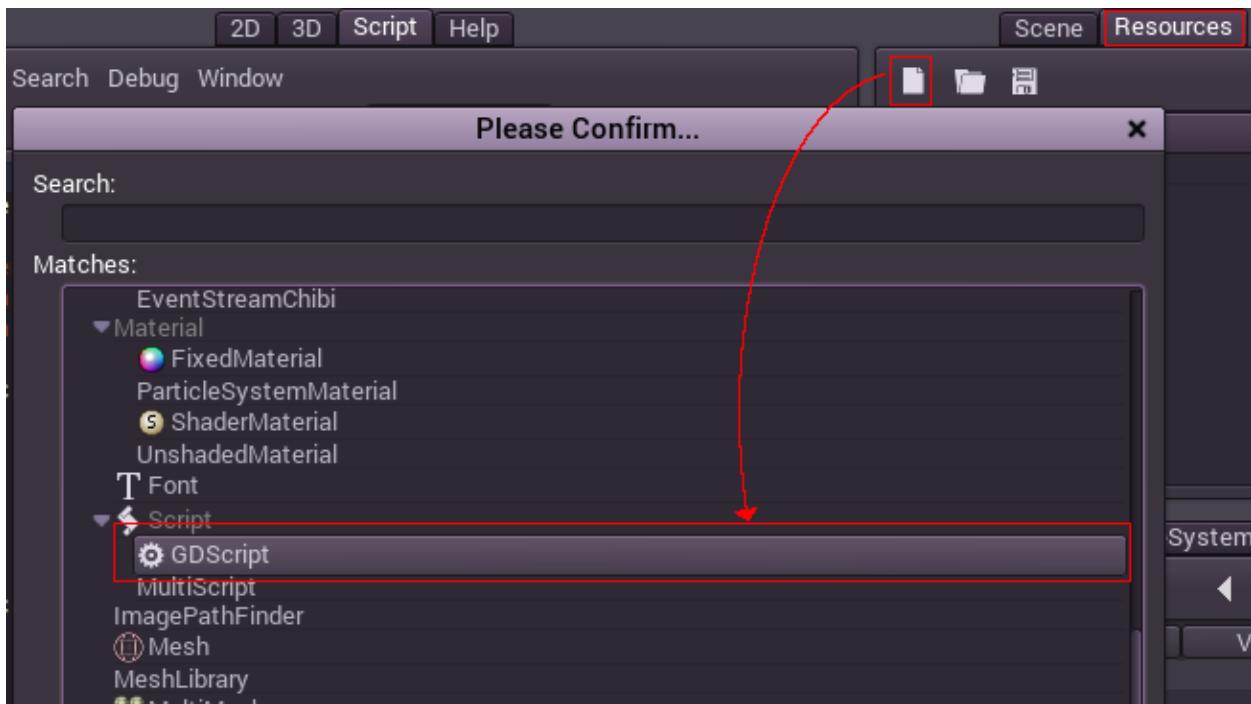
This short tutorial will explain how to make a scene switcher by using autoload. For simple scene switching, the `SceneTree.change_scene` method suffices (described [[Scene Main Loop]]), so this method is for more complex behaviors when switching scenes.

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

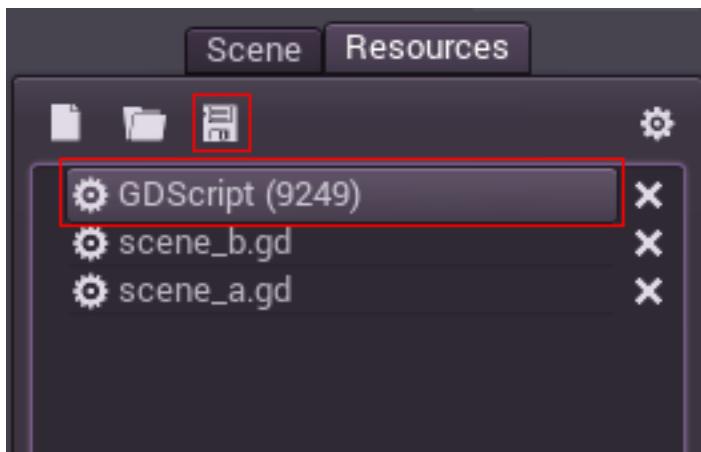
Two scenes are present, `scene_a.scn` and `scene_b.scn` on an otherwise empty project. Each are identical and contain a button connected to a callback for going to the opposite scene. When the project runs, it starts in `scene_a.scn`. However, this does nothing and pressing the button does not work.

### 1.13.4 global.gd

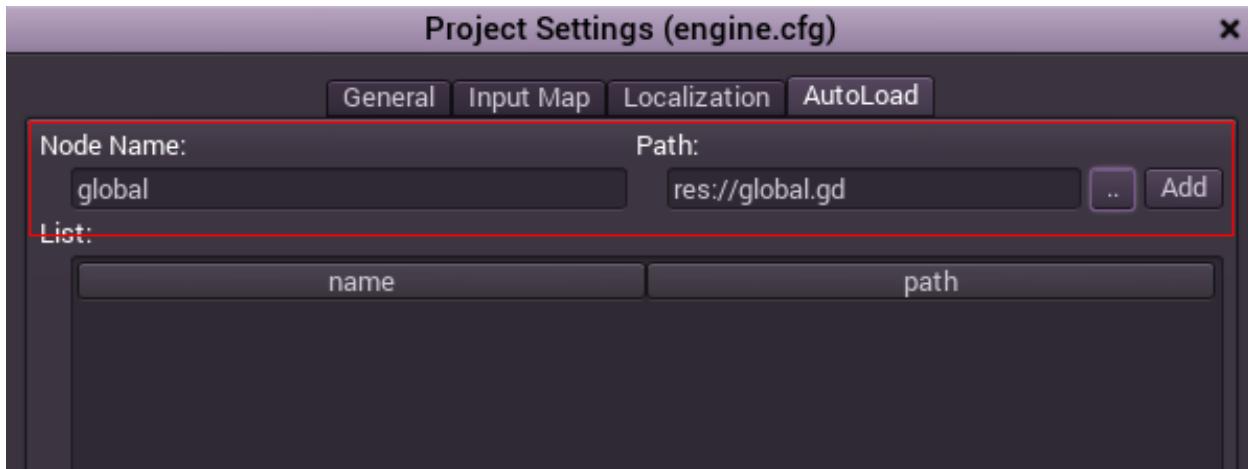
First of all, create a `global.gd` script. The easier way to create a resource from scratch is from the resources tab:



Save the script to a file global.gd:



The script should be opened in the script editor. Next step will be adding it to autoload, for this, go to: Scene [STRIKEOUT:> Project Settings]> AutoLoad and add a new autoload with name “global” that points to this file:



Now, when any scene is run, the script will be always loaded.

So, going back to it, In the `_ready()` function, the current scene will be fetched. 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.

Also, make sure that `global.gd` extends from `Node`, otherwise it won't be loaded.

```
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, is the function for changing scene. This function will erase the current scene and replace it by the requested one.

```
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 an immediate 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`:

```

#add to scene_a.gd

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

```

and

```

#add to scene_b.gd

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

```

Finally, by running the project it's possible to switch bewtween both scenes y pressing the button!

(To load scenes with a progress bar, check out the next tutorial, [[Background Loading]])



---

## Engine

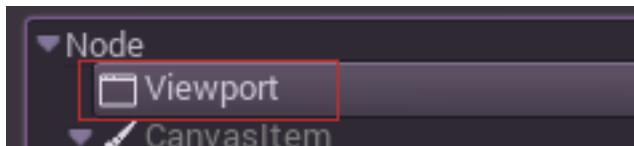
---

## 2.1 Scene, input & viewports

### 2.1.1 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:

```
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:

```
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 renderer 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:

```
# 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:

```
var capture = viewport.get_screen_capture()
```

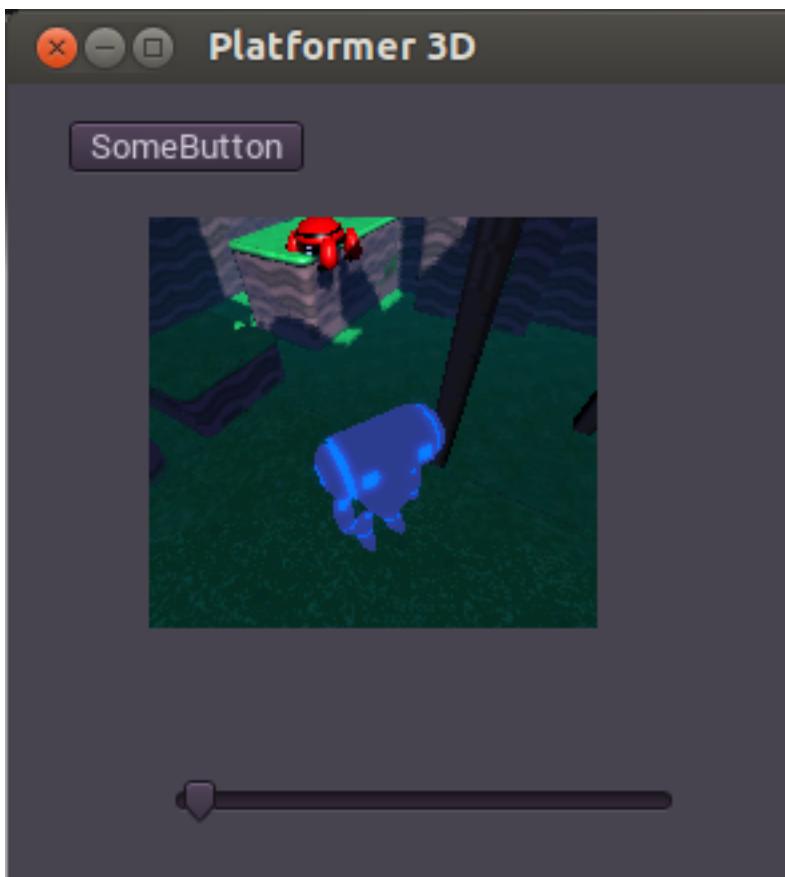
If the returned image is empty, capture still didn't happen, wait a little more, as this API is asynchronous.

## 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 it's 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):

```
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).

Make sure to check the [viewport demos!](#) Viewport folder in the demo.zip available to download, or <https://github.com/okamstudio/godot/tree/master/demos/viewport>

## 2.1.2 Screen Scaling & 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.

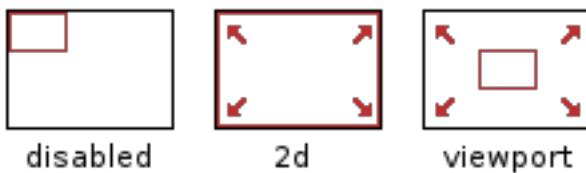
## Stretch Settings

Stretch settings are located in the project settings, it's just a bunch of configuration variables that provide several options:

Property	Value
<b>display</b>	
emulate_touchscreen	On
stretch_mode	viewport
stretch_aspect	keep
driver	GLES2
TNT_width	800

## 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.



## 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 ([[Gui Repositioning]]).
- **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).

## 2.1.3 InputEvent

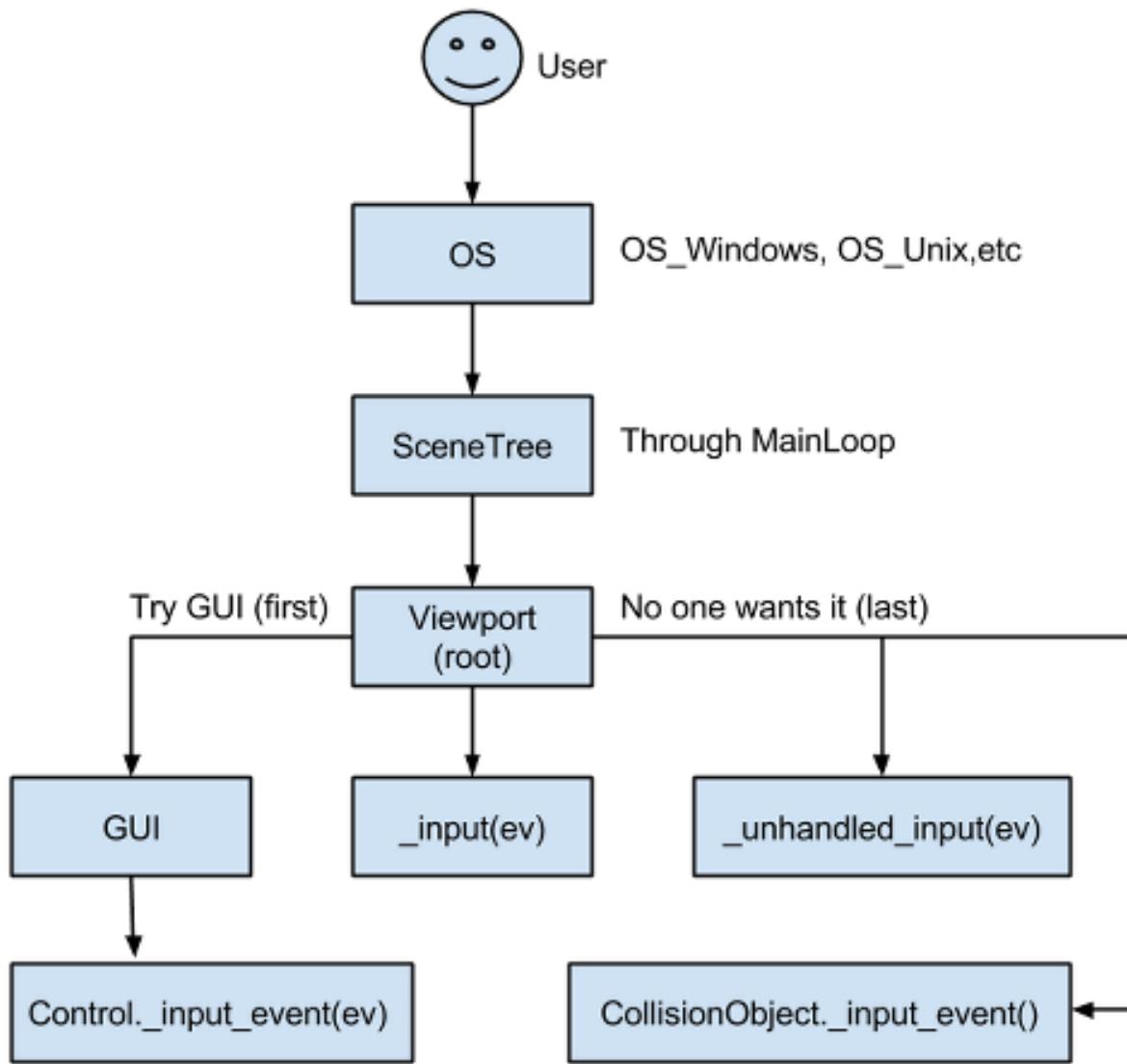
### What is it?

Managing input is usually complex, no matter the OS or platform. To ease this a little, a special built-in type is provided, [[API: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 [[API::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” [[API:Viewport]] (the first node of the scene tree). Viewport does quite a lot of stuff with the received input, in order:



1. First, it will try to feed the input to the GUI, and see if any control can receive it. If so, the [[API:Control]] will be called the virtual function [[API: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 [[API:Control.accept\_event()]] and the event will not spread any more.
2. If the GUI does not want the event, the standard \_input function will be called in any node with input processing enabled (enable with [[API:Node.set\_process\_input()]]) and override [[API:Node.\_input()]]. If any function consumes the event, it can call [[API:SceneTree.set\_input\_as\_handled()]], 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 [[API:Node.set\_process\_unhandled\_input()]]) and override [[API: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.
4. If no one wanted the event so far, and a [[API:Camera]] is assigned to the Viewport, a ray to the physics world (in the ray direction from the click) will be casted. If this ray hits an object, it will call the [[API:CollisionObject.\_input\_event()]] function in the relevant physics object (bodies receive this callback by default, but areas do not. This can be configured through [[API:Area]] properties).
5. Finally, if the event was unhandled, it will be passed to the next Viewport in the tree, or it will be ignored.

## Anatomy of an InputEvent

`[[API: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.

```
# 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:

Event	Type Index	Description
<code>[[API:InputEvent]]</code>	NONE	Empty Input Event
<code>[[API:InputEventKey]]</code>	KEY	Contains a scancode and unicode value, as well as modifiers
<code>[[API:InputEventMouse]]</code>	MOUSE_BUTTON	Contains click information, such as button, modifiers, etc.
<code>[[API:InputEventMouse]]</code>	MOUSE_MOTION	Contains motion information, such as relative, absolute positions and speed.
<code>[[API:InputEventJoystick]]</code>	STICK_MOTION	Contains Joystick/Joypad analog axis information.
<code>[[API:InputEventJoystick]]</code>	STICK_BUTTON	Contains Joystick/Joypad button information.
<code>[[API:InputEventScreen]]</code>	SCREEN_TOUCH	Contains multi-touch press/release information. (only available on mobile devices)
<code>[[API:InputEventScreen]]</code>	SCREEN_DRAG	Contains multi-touch drag information. (only available on mobile devices)
<code>[[API:InputEventAction]]</code>	SCREEN_ACTION	Contains a generic action. These events are often generated by the programmer as feedback. (more on this below)

## 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 (ie: 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. If you read the [[Tutorial 2D]], there is an explanation on how does the action editor work.

Any event has the methods `[[API:InputEvent.is_action()]]`, `[[API:InputEvent.is_pressed()]]` and `[[API:InputEvent.is_echo()]]`.

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). `SceneTree` (derived from `MainLoop`) has a method for this: `[[API:MainLoop.input_event(ev)]]`. You would normally use it like this:

```
var ev = InputEvent()
ev.type=InputEvent.ACTION
# set as move_left, pressed
```

```
ev.set_as_action("move_left", true)
# feedback
get_tree().input_event(ev)
```

## InputMap

Customizing and re-mapping input from code is often desired. If your whole workflow depends on actions, the [[API: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 sees best fit.

### 2.1.4 Mouse & 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, make not as much sense outside of that scope.

[STRIKEOUT:The only way to reliably obtain this information is by using functions such as:]

**This method is no longer supported:** It was too confusing and caused errors for users making 2D games. Screen would stretch to different resolutions and input would stop making sense. Please use the “\_input” function

```
OS.get_video_mode_size()
Input.get_mouse_pos()
```

However, this is discouraged for pretty much any situation. Please do not use these functions unless you really know what you are doing.

#### Viewport Display Coordinates

Godot uses viewports to display content, and viewports can be scaled by several options (see [[tutorial\_multires]] tutorial). Use, then, the functions in nodes to obtain the mouse coordinates and viewport size, for example:

```
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()
```

## 2.2 Filesystem

### 2.2.1 Version Control & 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:

```
myproject/game/engine.cfg  
myproject/game/scenes/house/house.scn  
myproject/game/scenes/house/sometexture.tex  
myproject/game/sound/door_open.smp  
myproject/game/sound/door_close.smp  
myproject/game/translations/sheet.en.xls  
myproject/game/translations/sheet.es.xls
```

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/texture.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_sceen.scn  
game/gui/options/options.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.

## 2.2.2 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.

## 2.2.3 Saving Your Game

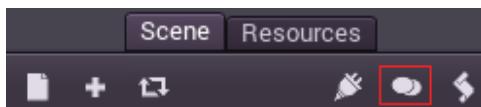
### Introduction

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:

```
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 file 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:

```
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 } 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.

```
# 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 filename and

parent 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 w...
    #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://Invasionsave.save", File.READ)
    while (!savegame.eof_reached()):
        currentline.parse_json(savegame.get_line())
        #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])
    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) 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.

## 2.2.4 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:

```
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:

```
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()
```

This is all! Thanks for your cooperation, citizen.

## 2.3 Internationalization

### 2.3.1 Internationalization

#### Introduction

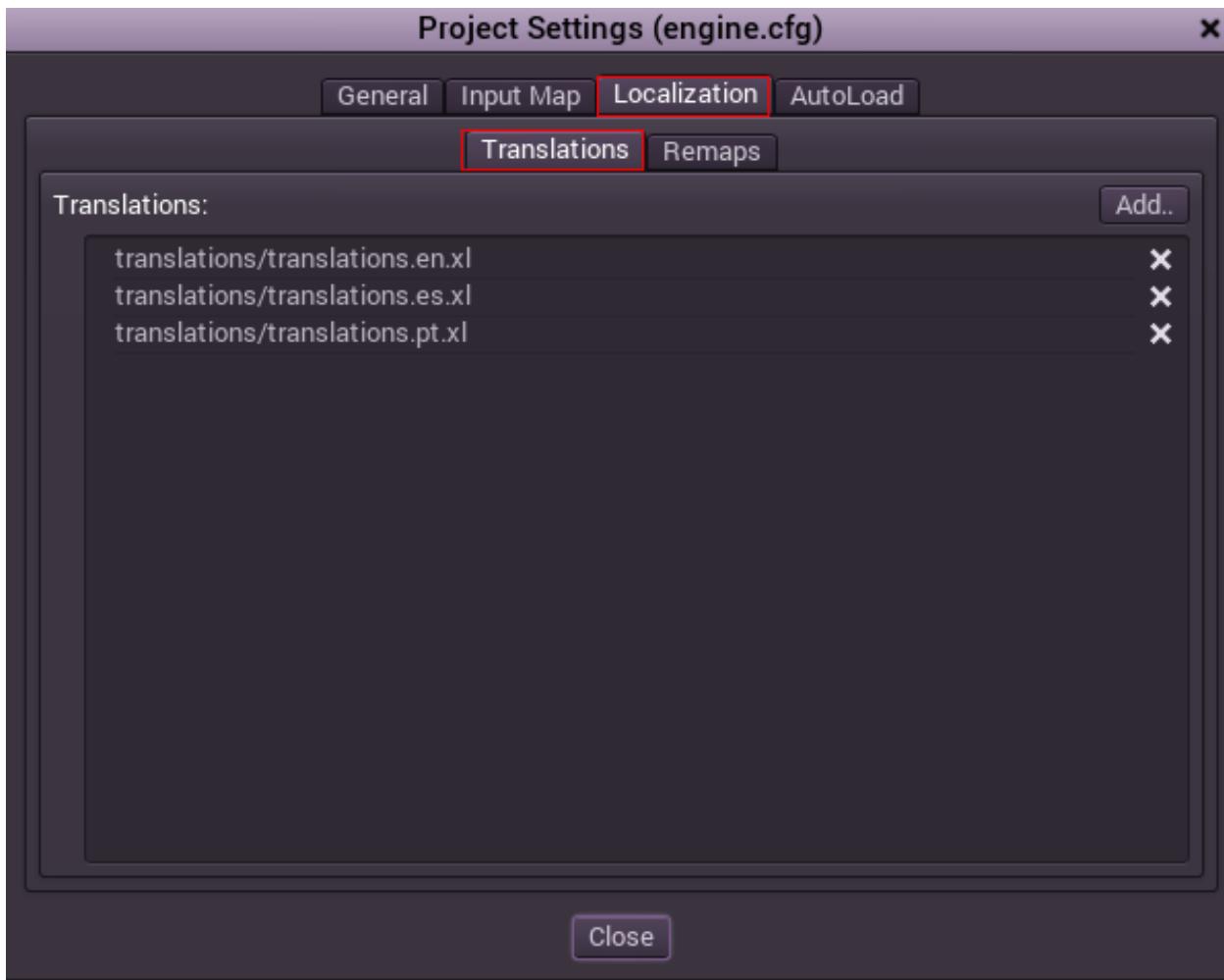
Sería excelente que el mundo hablara solo un idioma. Unfortunately for us developers, that is not the case. While not generally a big requirement when developing indie or niche games, it is also very common that games going into a more massive market require localization.

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 [[Import Translation]] tutorial, so this one could be seen more like a follow up to that one.

#### Configuring the Imported Translation

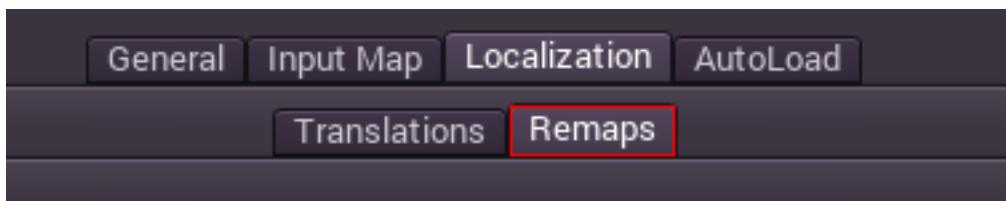
The translations can get updated and re-imported when they change, but they still have to be added to the project. This is done in Scene [STRIKEOUT:> Project Settings]> Localization:



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](#). 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:

```
level.set_text(tr("LEVEL_5_NAME"))
status.set_text(tr("GAME_STATUS_"+str(status_index)))
```

## Making Controls Resizable

The same text in different languages can vary greatly in length. For this, make sure to read the tutorial on [[GUI Repositioning]], as having dynamically adjusted control sizes may help. [Containers](#) 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.

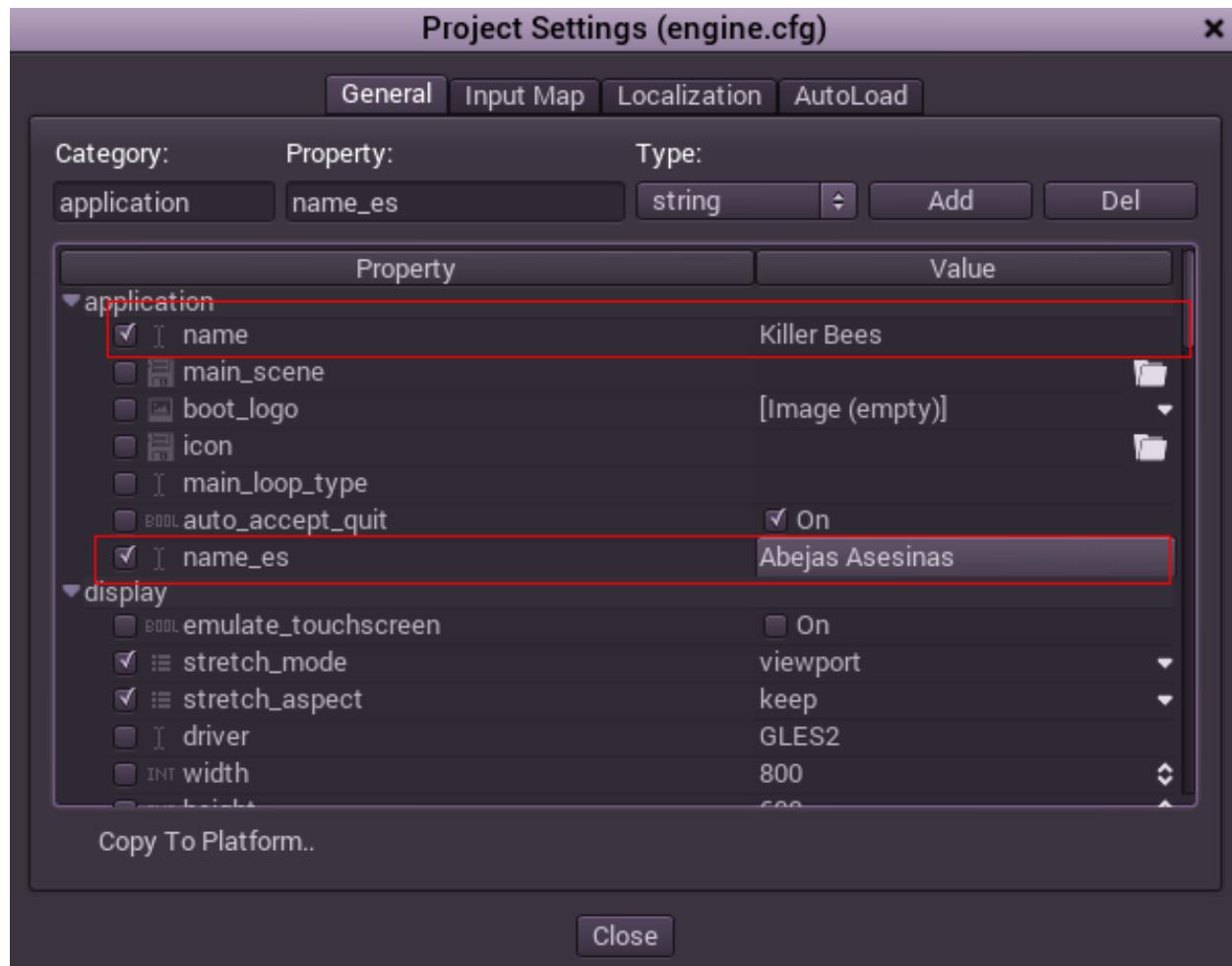
## Command Line

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. In the project settings dialog, create a new setting application/name and append it the locale identifier. For example:



As always, If you don't know the code of a language or zone, [check the list](#).

## 2.4 Game flow

### 2.4.1 Pausing a Game

#### Pause?

In most games, it is always desirable to, at some point, interrupt the game to do something else. Be it taking a break, to 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 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:

```
get_tree().set_pause(true)
```

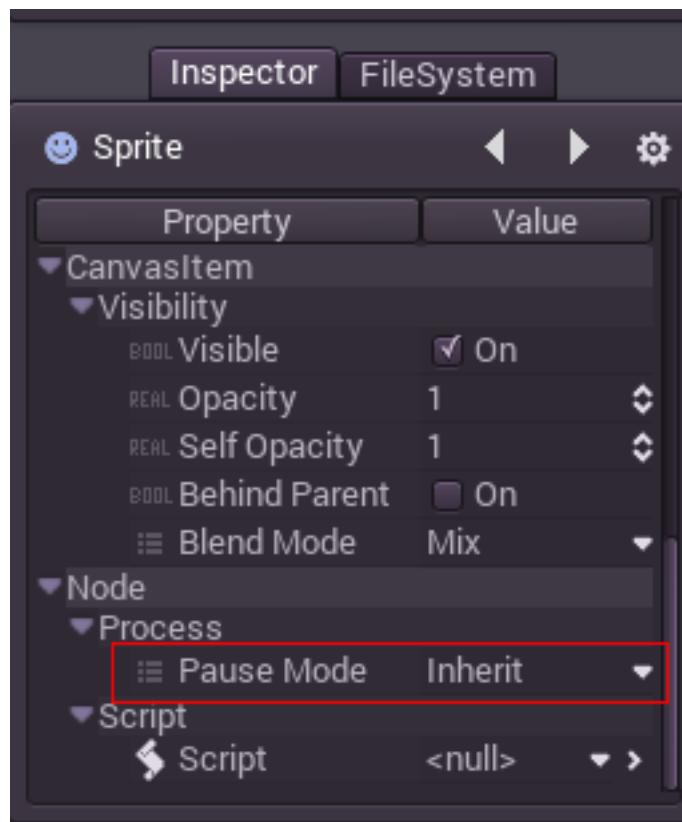
Doing so will have the following behavior:

- 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:



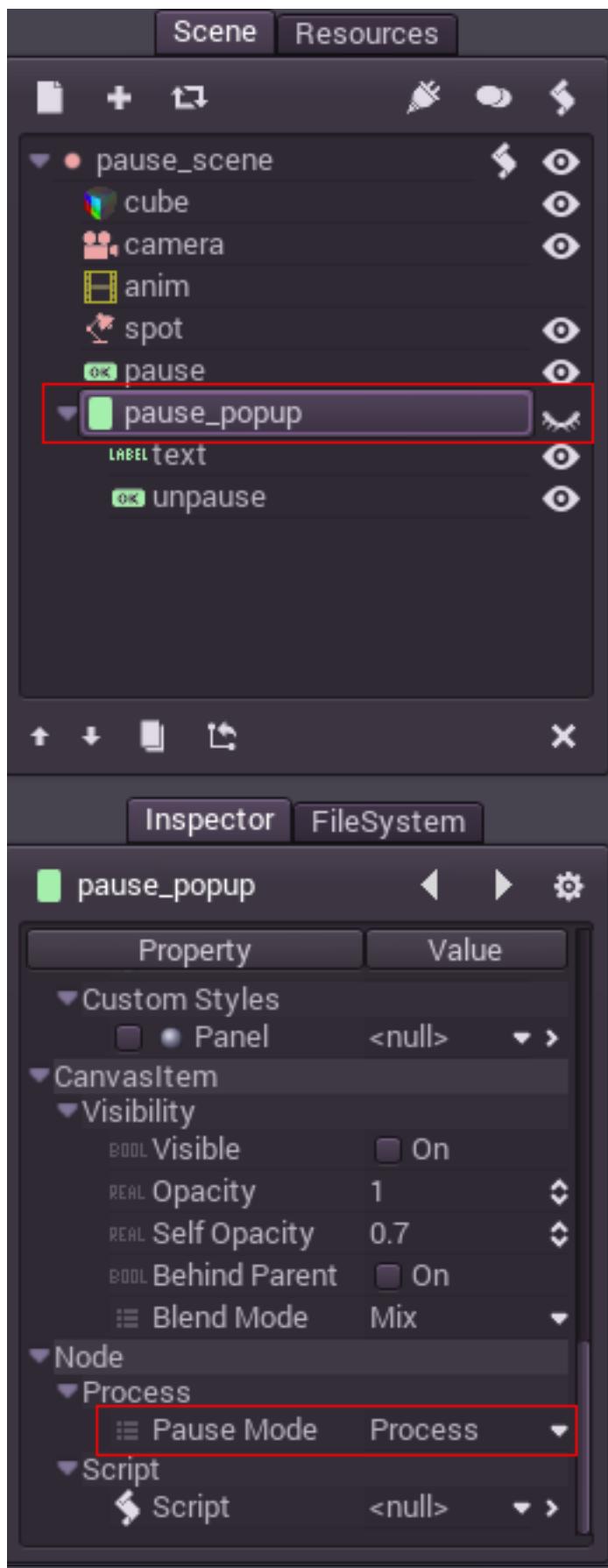
By default all nodes have this property in the “Inherit” state. This means, that they will only process (or not) depending on what this same property is set on the parent node. If the parent is set to “Inherit”, then the grandparent will be checked and so on. Ultimately, if a state can’t be found in 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). Paused or not 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 it's 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.

```
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:

```
func _on_pause_popup_close_pressed():
    get_node("pause_popup").hide()
    get_tree().set_pause(false)
```

And that should be all!

## 2.4.2 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

```
Ref ResourceLoader::load_interactive(String p_path);
```

This method will give you a `ResourceInteractiveLoader` that you will use to manage the load operation.

#### Polling

```
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:

```
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)

```
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

```
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 [[[https://github.com/okamstudio/godot/wiki/tutorial\\_singletons#scene-switcher](https://github.com/okamstudio/godot/wiki/tutorial_singletons#scene-switcher)]] example.

First we setup some variables and initialize the

`current_scene`

with the main scene of the game:

```
var loader
var wait_frames
var time_max = 100 h1. 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 `_process` callback. It also starts a “loading” animation, which can show a progress bar or loading screen, etc.

```
func goto_scene(path): h1. 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 we 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.

```
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 frame

        # poll your loader
        var err = loader.poll()

        if err == ERR_FILE_EOF: # load finished
            var resource = loader.get_resource()
            loader = null
            set_new_scene(resource)
            break
        elif err == OK:
            update_progress()
        else: h1. error during loading
            show_error()
            loader = null
            break
```

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.

```

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).

### Don't block the main thread during the call to poll

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](#). 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:

```
# 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
# 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
# of the queue, pausing the load of any other resource

# to check progress
if queue.is_ready("res://level_1.xml"):
    show_new_level(queue.get_resource("res://level_1.xml"))
else:
    update_progress(queue.get_process("res://level_1.xml"))

# 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. Find me on IRC (punto on irc.freenode.net) or e-mail me ([punto@okamstudio.com](mailto:punto@okamstudio.com)) for help.

### 2.4.3 Handling Quit Request

#### 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):

```
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:

```
get_tree().set_auto_accept_quit(false)
```



---

## 2D tutorials

---

### 3.1 Graphics

#### 3.1.1 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, 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, points, 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?

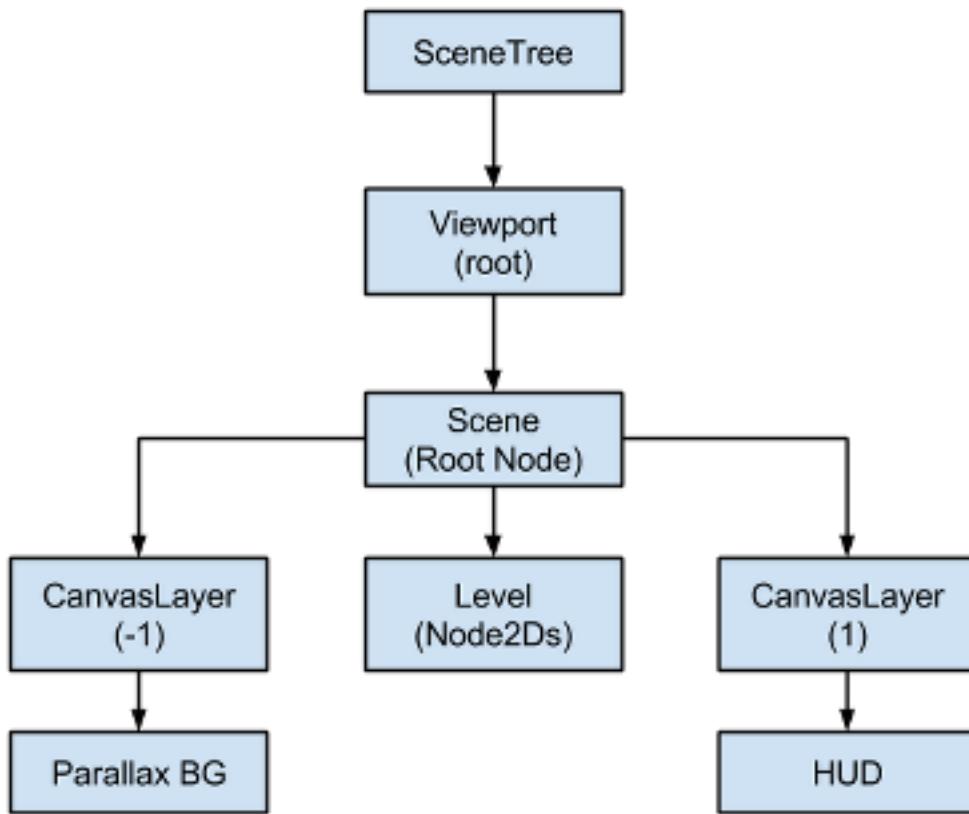
##### CanvasLayers

The answer is [CanvasLayer](#), which is a node that adds a separate 2D rendering layer for all it’s 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:



`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. In the future, nodes will also have a priority or sub-layer index which should aid for this.

### 3.1.2 Viewport & 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](#).

By default, nodes are drawn in Layer 0, in the built-in canvas. To put nodes in a different layer, a [CanvasLayer](#) node can be used. This was covered in the previous tutorial anyway, just refreshing.

## 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.

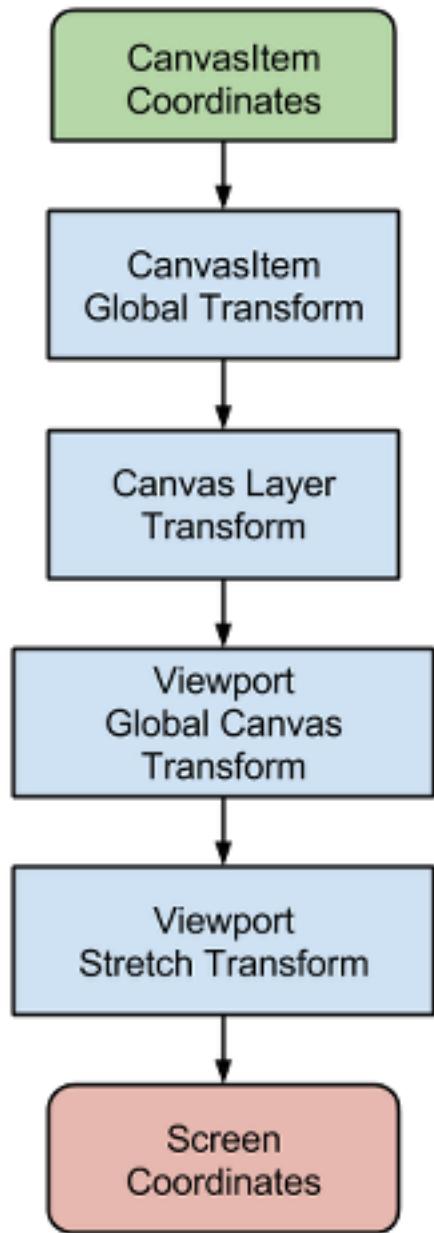
## Stretch Transform

Finally, viewports have a *Stretch Transform*, which is used when resizing or stretching the screen. This transform is used internally by the [[Tutorial Multires]], but can also be requested to the viewport.

Input events received in the [Node.\\_input\\_event\(ev\)](#) callback are multiplied by this transform, but lack the ones above. To convert InputEvent coordinates to local CanvasItem coordinates, the [CanvasItem.make\\_input\\_local\(ev\)](#) 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:

```
|Type: | Transform |
|CanvasItem| CanvasItem.get\_global\_transform\(\) |
|CanvasLayer| CanvasItem.get\_canvas\_transform\(\) |
|CanvasLayer+GlobalCanvas+Stretch| CanvasItem.get\_viewport\_transform\(\) |
```

Finally then, to convert a CanvasItem local coordinates to screen coordinates, just multiply in the following order:

```
var screen_coord = get_viewport_transform() + ( get_global_transform() + local_pos )
```

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:

```
var local_pos = Vector2(10,20) # local to Control/Node2D
var ie = InputEvent()
ie.type=InputEvent.MOUSE_BUTTON
ie.button_index=1 #left click
ie.pos = get_viewport_transform() + ( get_global_transform() + local_pos )
get_tree().input_event(ie)
```

### 3.1.3 Custom Drawing in 2D

#### Why?

Godot has nodes to draw sprites, polygons, particles, and all sort of stuff. For far most cases this is enough, but not always. If something desired is not supported, and before crying in fear, angst and range 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

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)
```

OK! This is basically it! Enjoy drawing your own nodes!

## 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).

### 3.1.4 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.

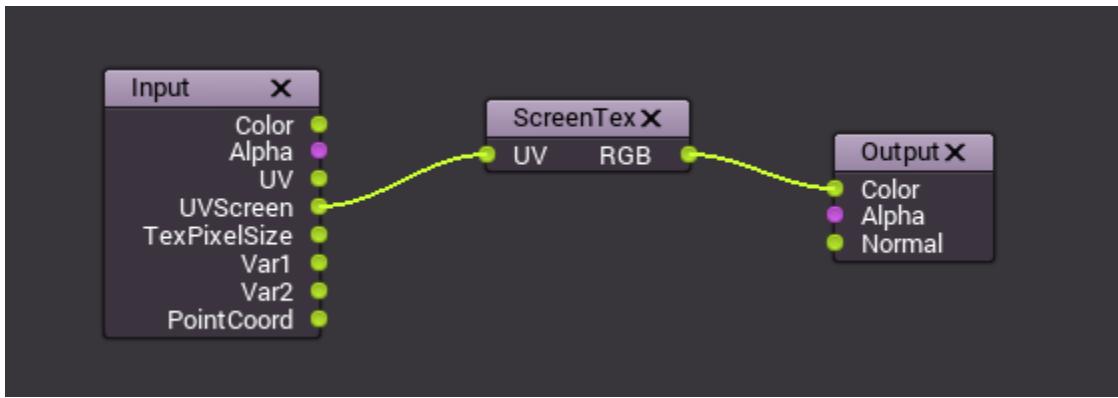
The workaround is to make a copy of the screen, or a part of the screen, to a back-buffer and then read from it while drawing. Godot provides a few tools that makes this process easy!

### TexScreen shader instruction.

Godot [[Shader]] 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);

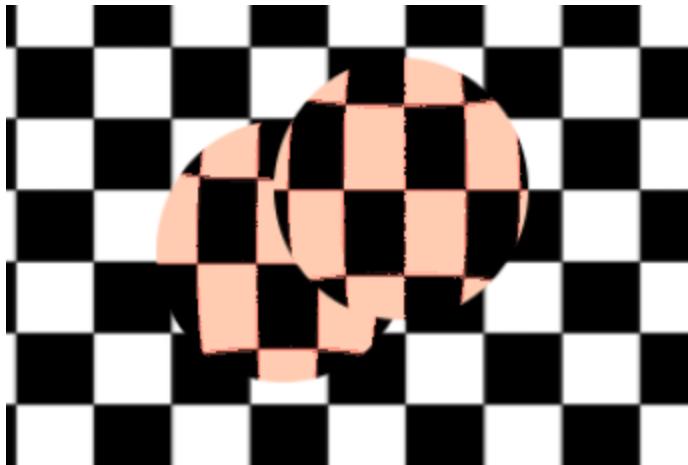
c.rgb = mix(vec3(0.0),c.rgb,brightness);
c.rgb = mix(vec3(0.5),c.rgb,contrast);
c.rgb = mix(vec3(dot(vec3(1.0),c.rgb)*0.33333),c.rgb,saturation);

COLOR.rgb=c;
```

### Behind The Scenes

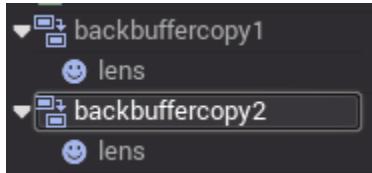
While this seems magical, it's not. The Texscreen instruction, when first found in a node that is about to be drawn, does a full-screen copy to a back-buffer. Subsequent nodes that use texscreen() in shaders will not have the screen copied for them, because this ends up being very inefficient.

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:

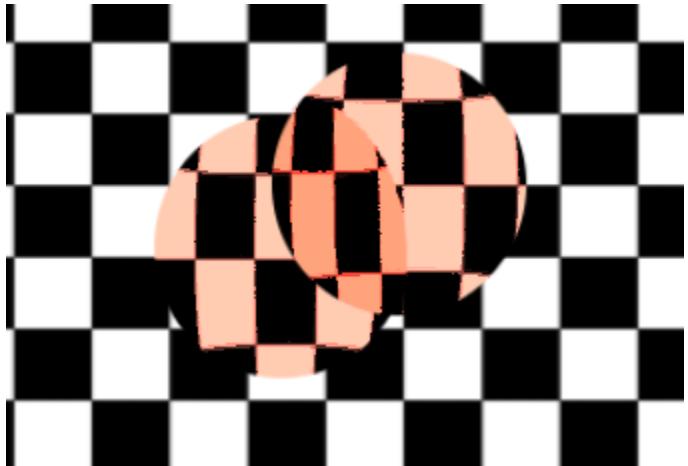


In the above image, the second sphere (top right) is using the same source for `texscreen()` as the first one below, so the first one “dissapears”, or is not visible.

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()`, 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 [STRIKEOUT: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!

### 3.1.5 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.

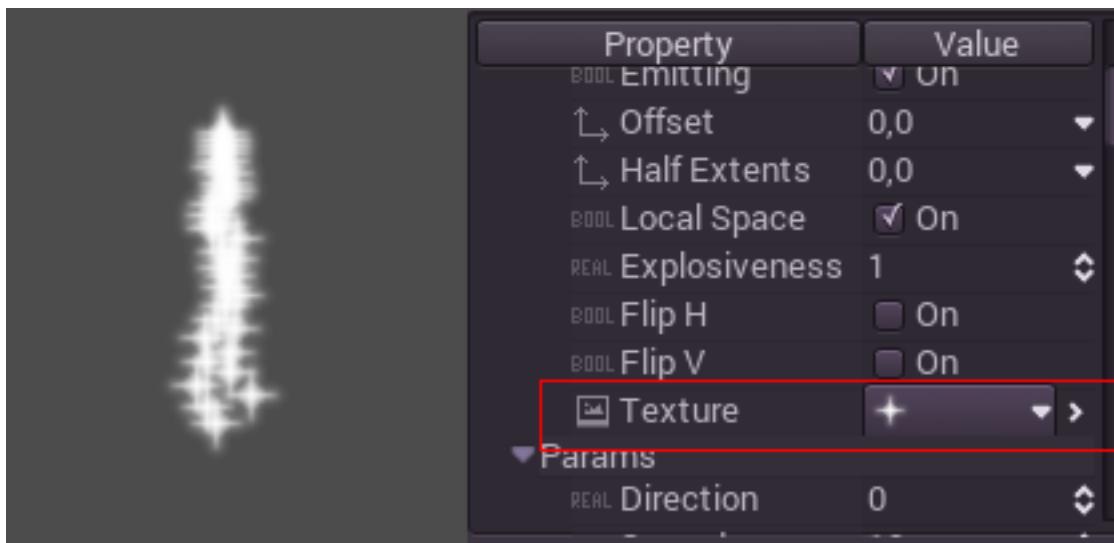
#### Particles2D

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 emitter 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. Combined 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 intial angle of the particle (in degress). This parameter is mostly useful randomized.

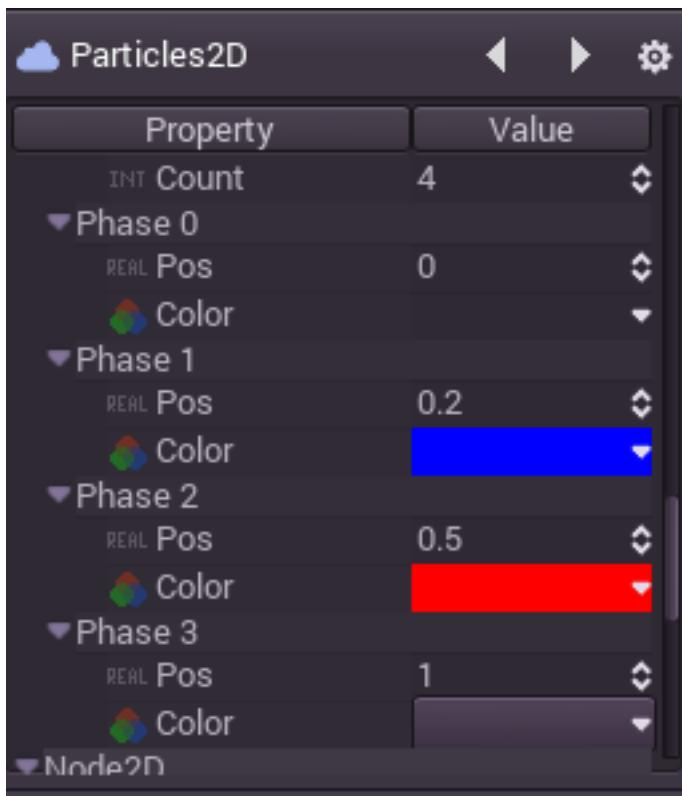
## Initial & Final Size

Determines the intial 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 alays 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:

```
initial_value = param_value + param_value*randomness
```

## 3.1.6 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 became 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 it's 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 face 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!

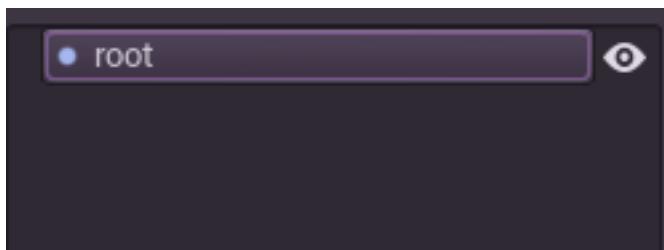
## Making of GBot!

For this tutorial, we will use as demo content the pieces of the [GBot](#) character, created by Andreas Esau.

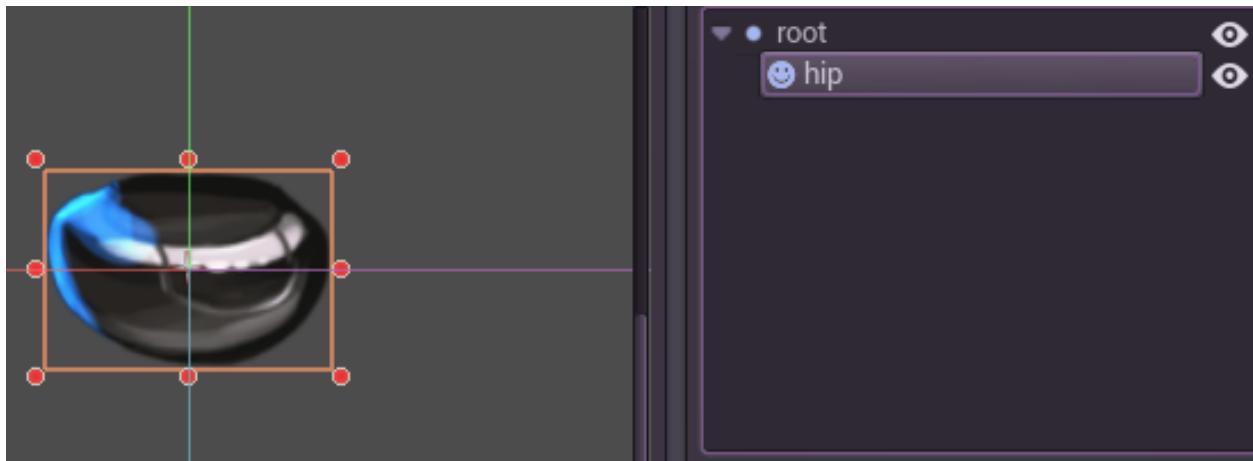
Get your assets attachment:[gbot\\_resources.zip](#).

## Setting up the Rig

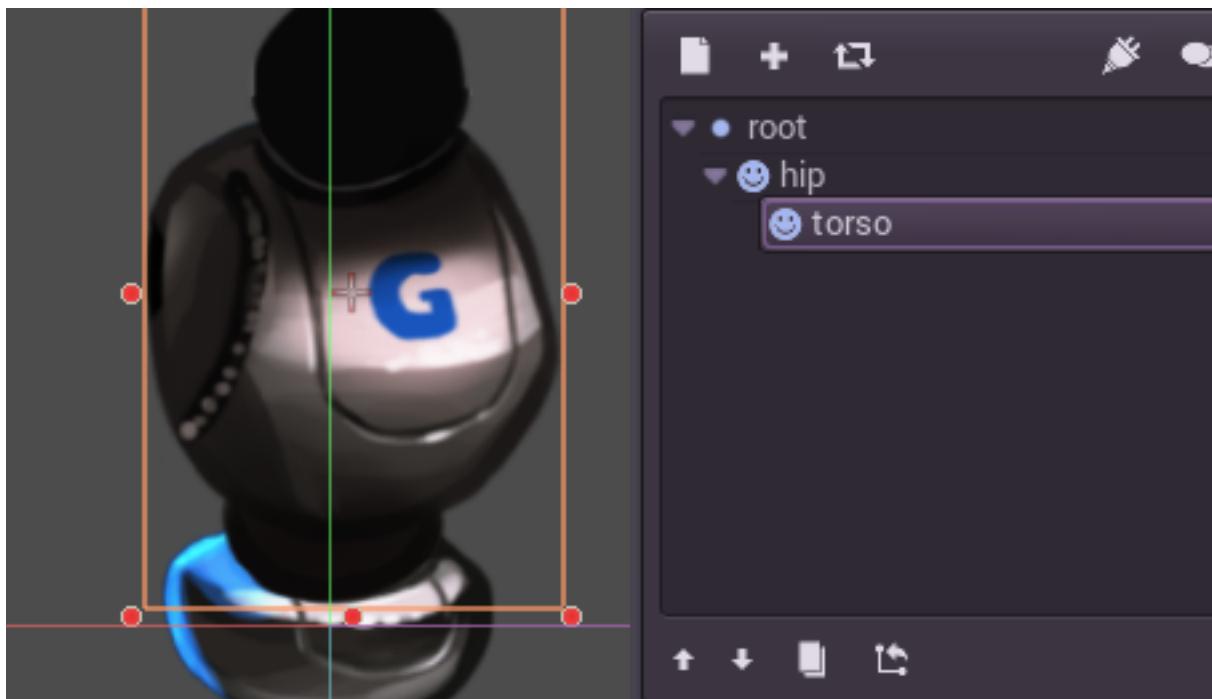
Create an empty Node2D as root of the scene, we'll work under it:



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:

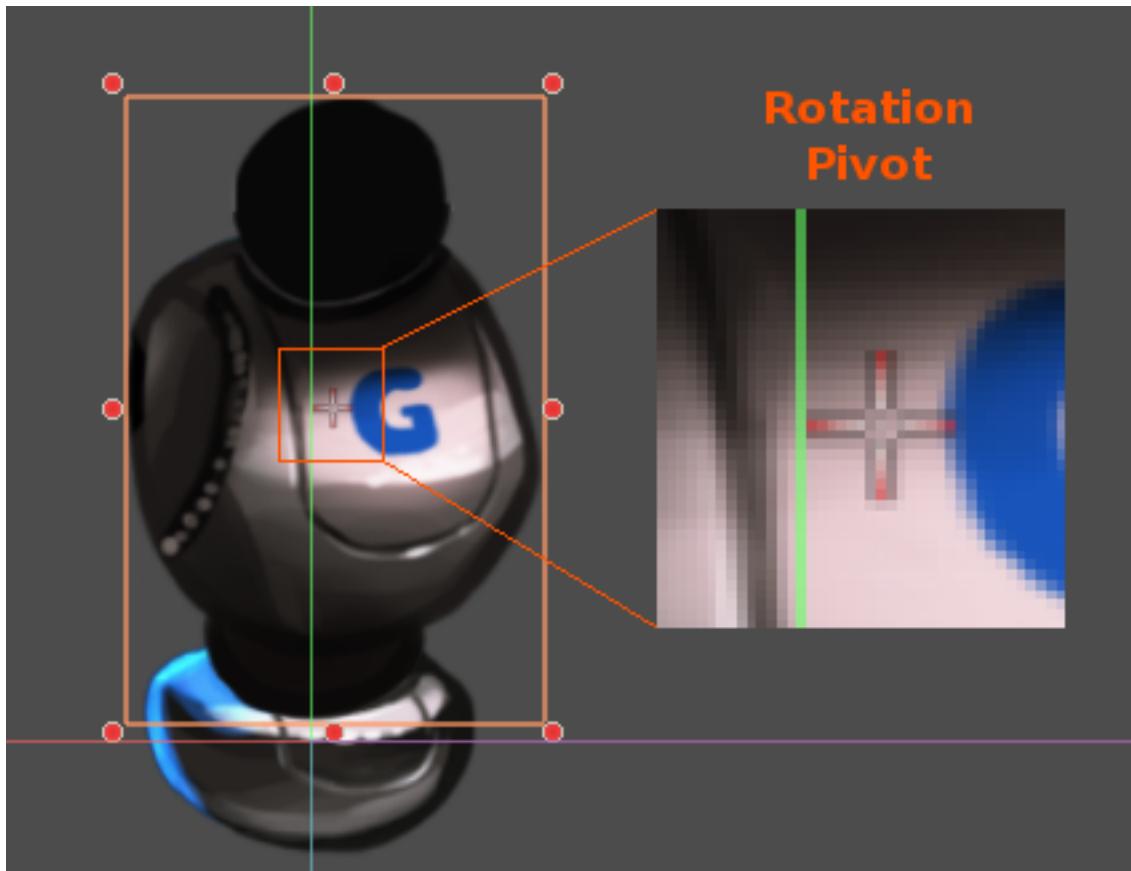


Next will be the torso. The torso needs to be a child of the hip, so create a child sprite and load the torso, later accommodate it properly:



This looks good. Let's try 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:



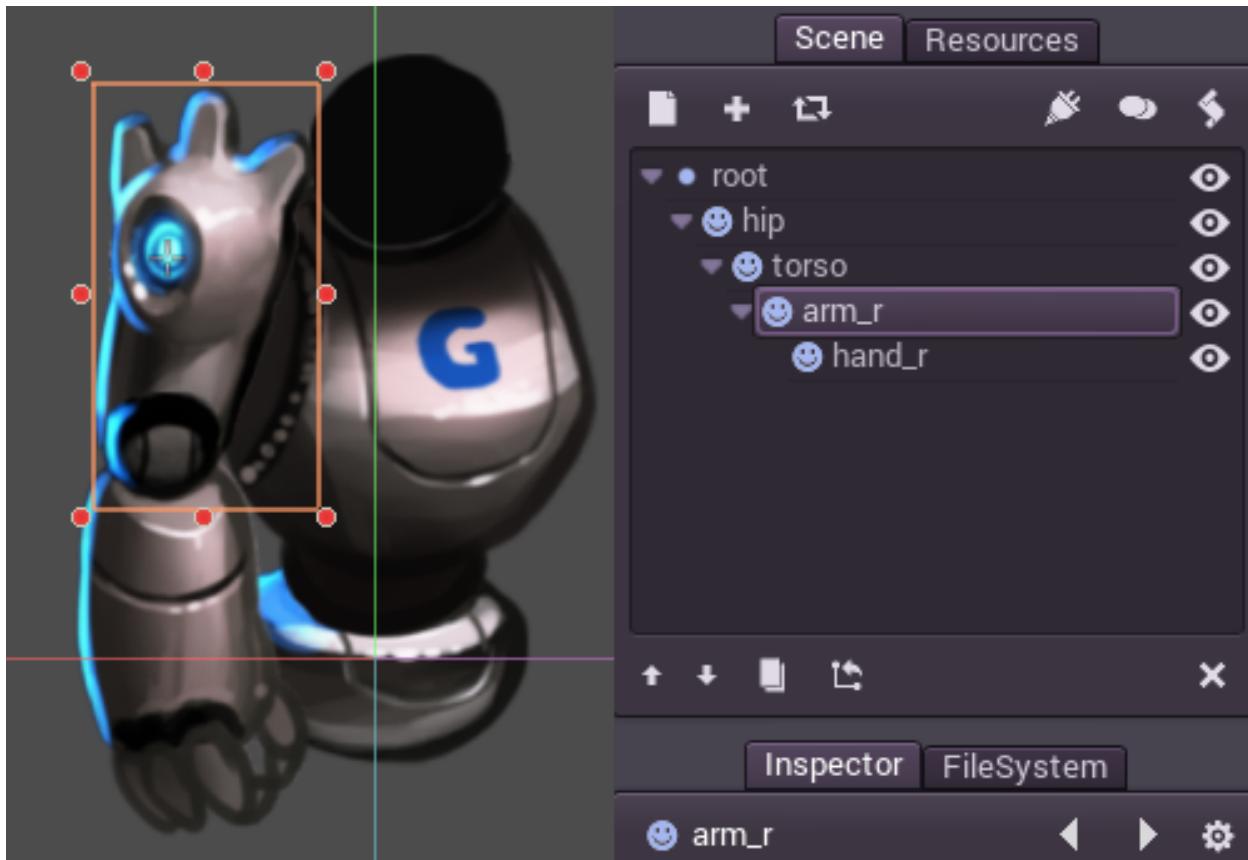
### Adjusting the Pivot

The Pivot can be adjusted by changing the *offset* property in the Sprite:

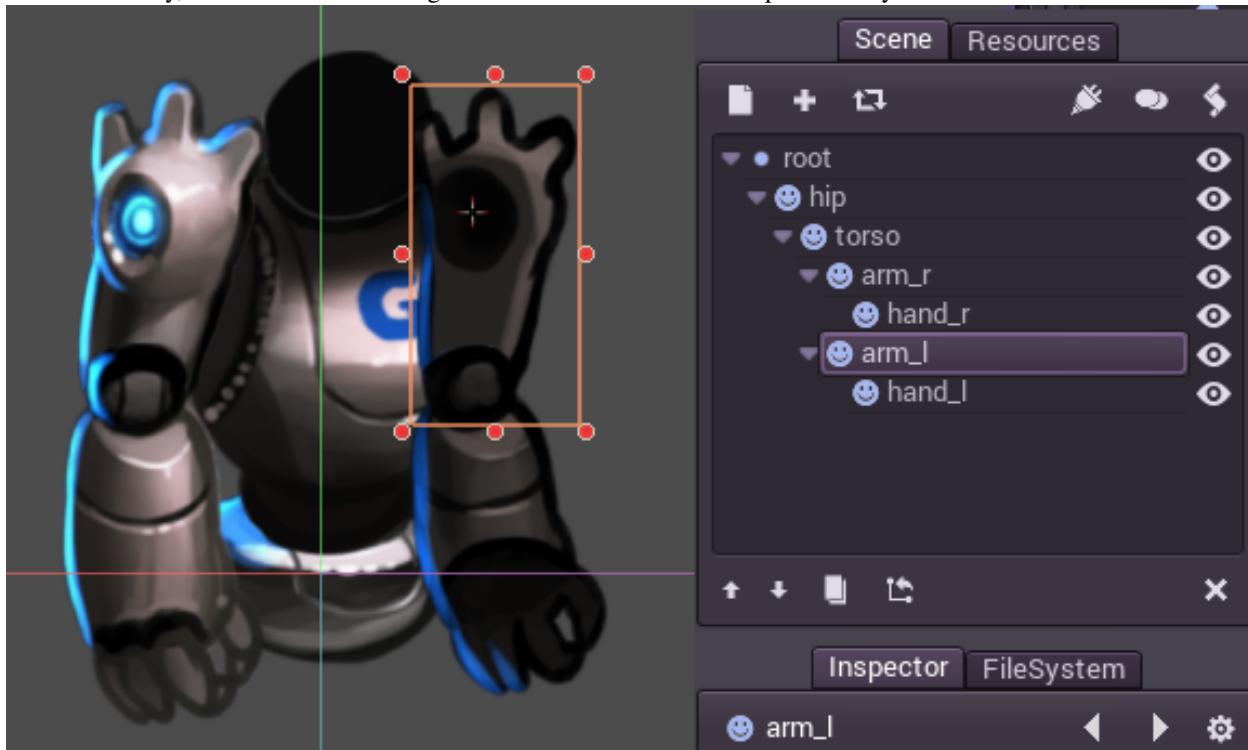


However, there is a way to do it more *visually*. Pick the object and move it normally. After the motion has begun and while the left mouse button is being held, press the “v” key *without releasing* the mouse button. Further motion will move the object around the pivot. This small tool allows adjusting the pivot easily. Finally, move the pivot to the right place:

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:



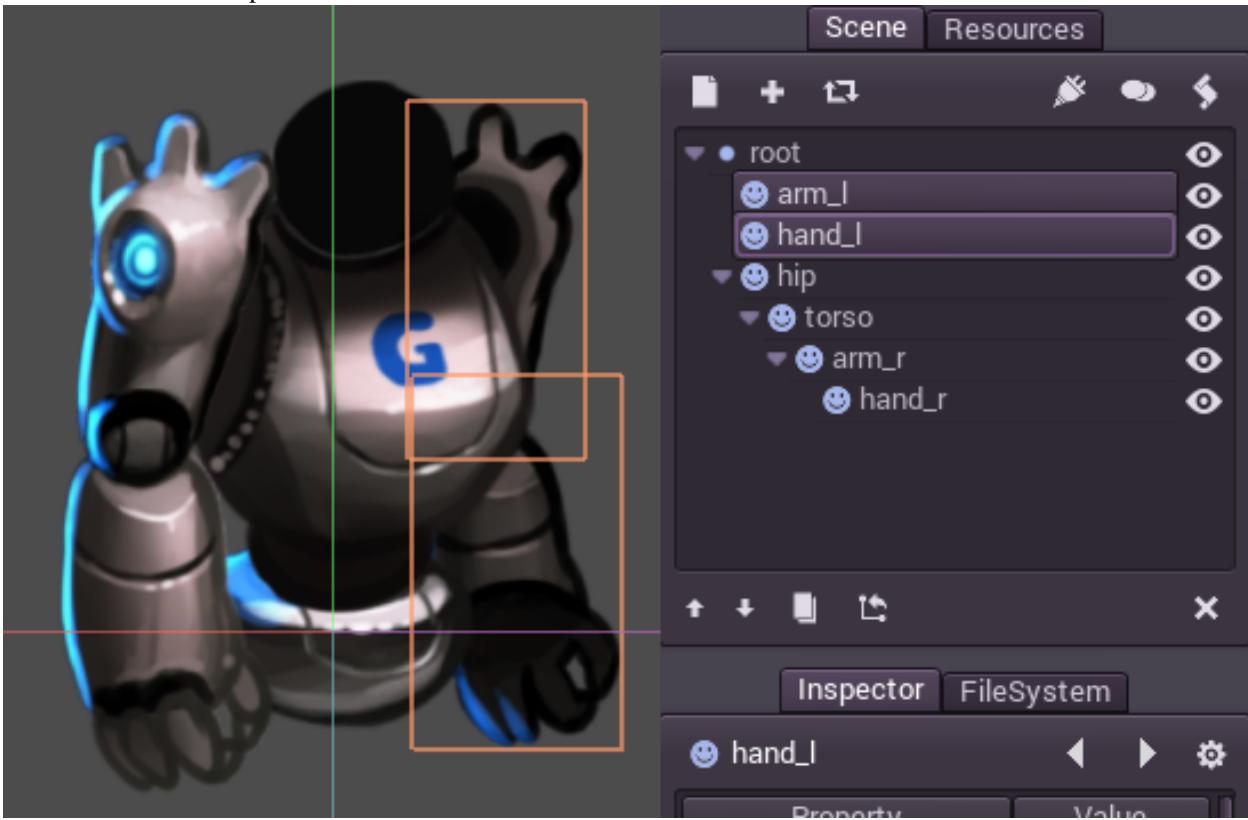
This seems easy, so continue with the right arm. The rest should be simple! Or maybe not:



Right. Remember your tutorials, Luke. In 2D, parent nodes appear below children nodes. Well, this sucks. It seems

Godot does not support cutout rigs after all. Come back next year, maybe for 1.2.. no wait. Just Kidding! It works just fine.

But how can this problem be solved? We want the whole torso to appear behind the hip and the torso. For this, we can move the nodes behind the hip:



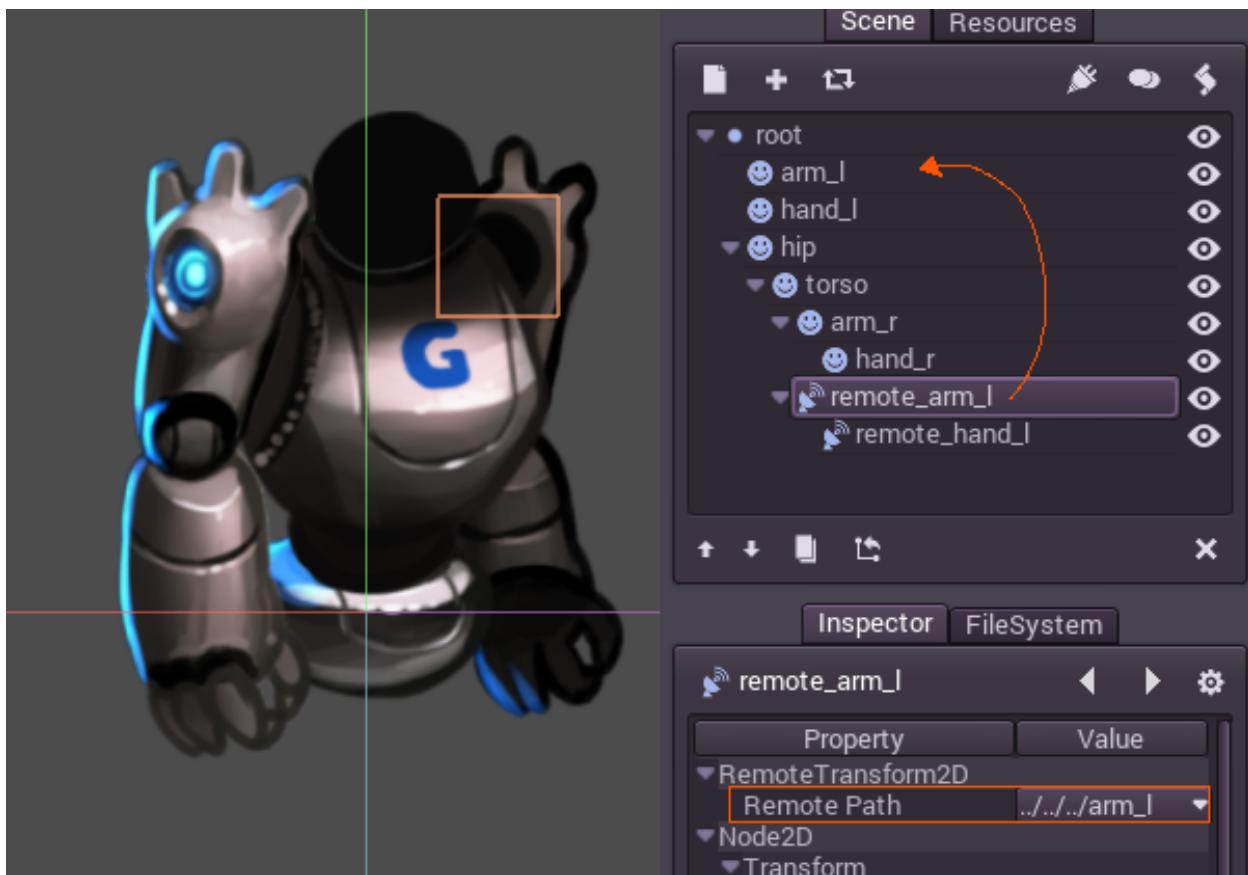
But then, we lose the hierarchy layout, which allows to control the skeleton like.. a skeleton. Is there any hope?.. Of Course!

### RemoteTransform2D Node

Godot provides a special node, [RemoteTransform2D](#). This node will transform nodes that are sitting somewhere else in the hierarchy, by copying it's transform to the remote node.

This enables to have a visibility order independent from the hierarchy.

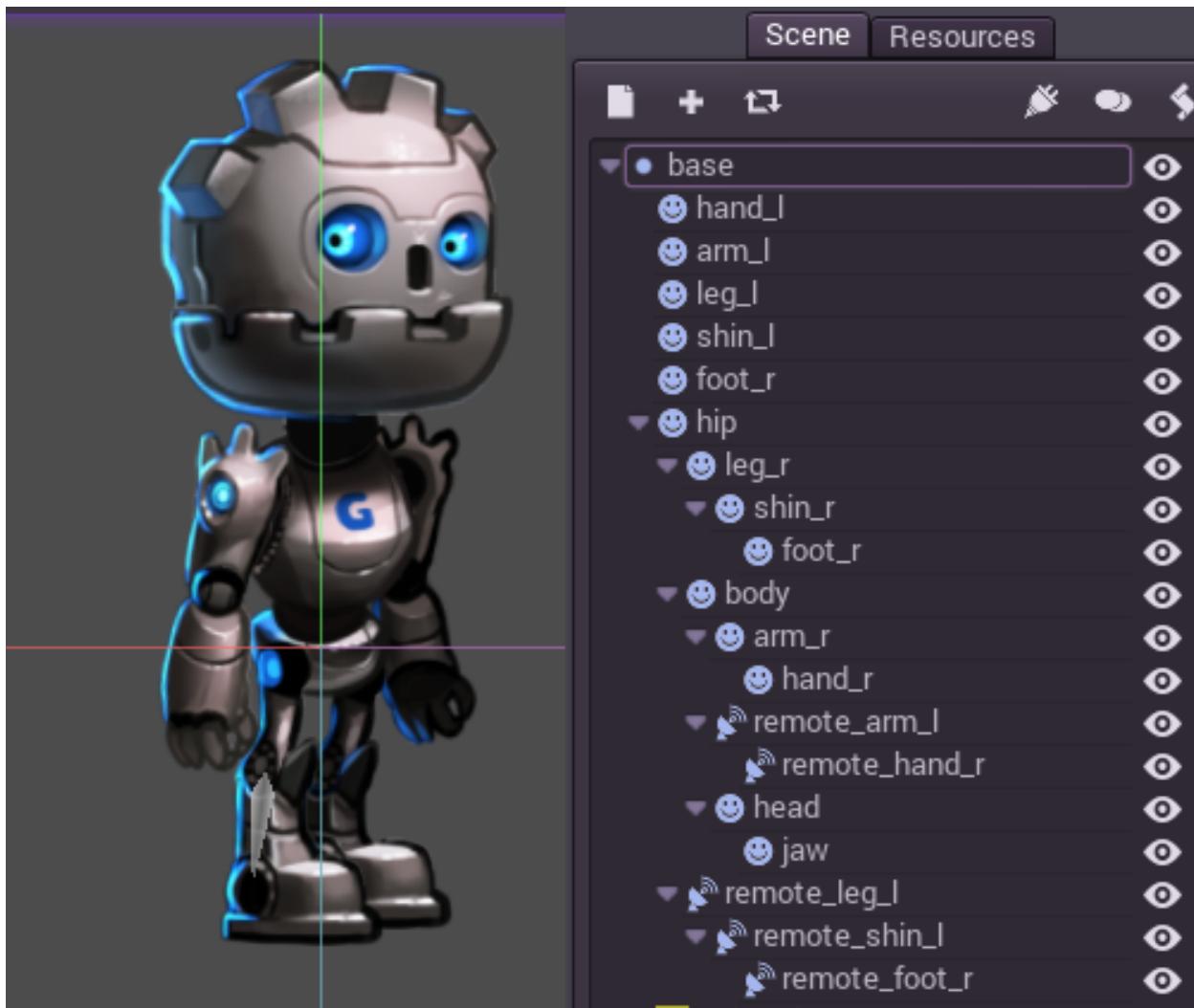
Simply create two more nodes as children from torso, remote\_arm\_l and remote\_hand\_l and link them to the actual sprites:



Moving the remote transform nodes will move the sprites, allowing 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 should be easy to animate, by selecting the nodes and rotating them you can animate forward kinematic (FK) efficiently.

For simple objects and rigs this is fine, however the following problems are common:

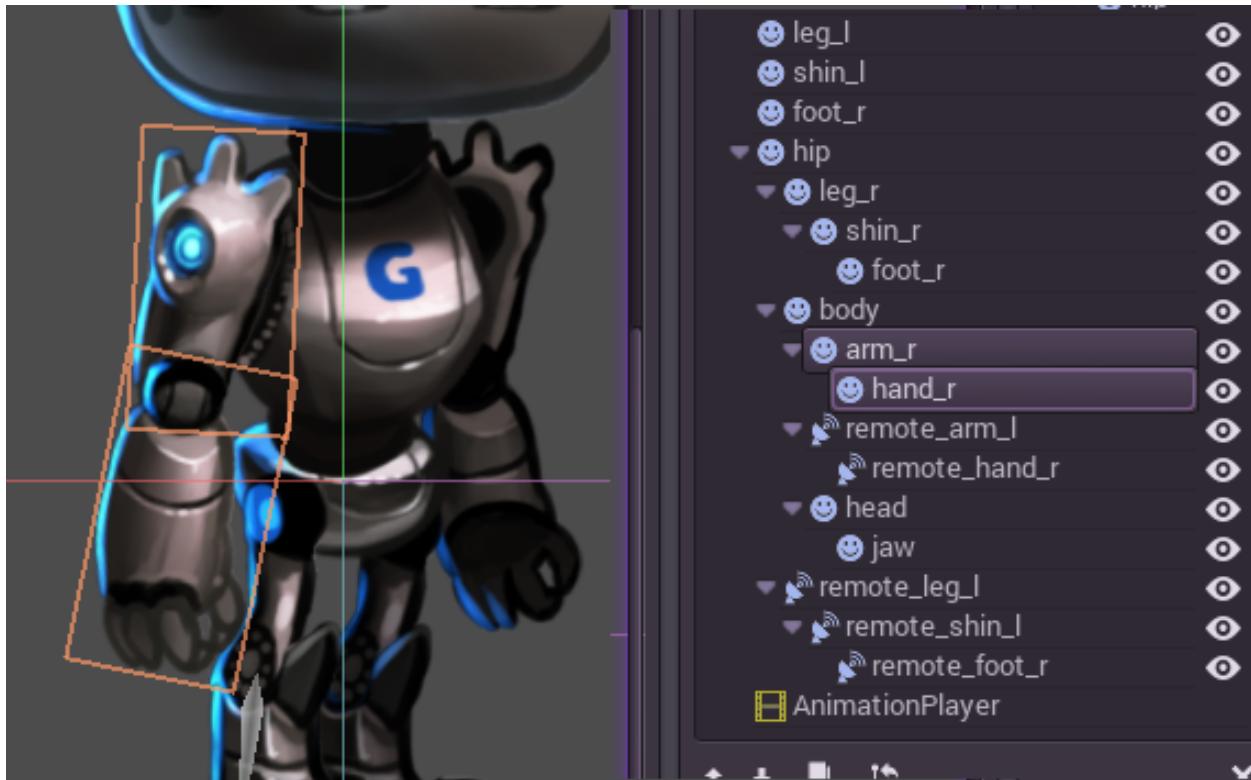
- Selecting sprites can become difficult for complex rigs, and the scene tree ends being used due to the difficulty of clicking over the proper sprite.
- Inverse Kinematics is often desired for extremities.

To solve these problems, Godot supports a simple method of skeletons.

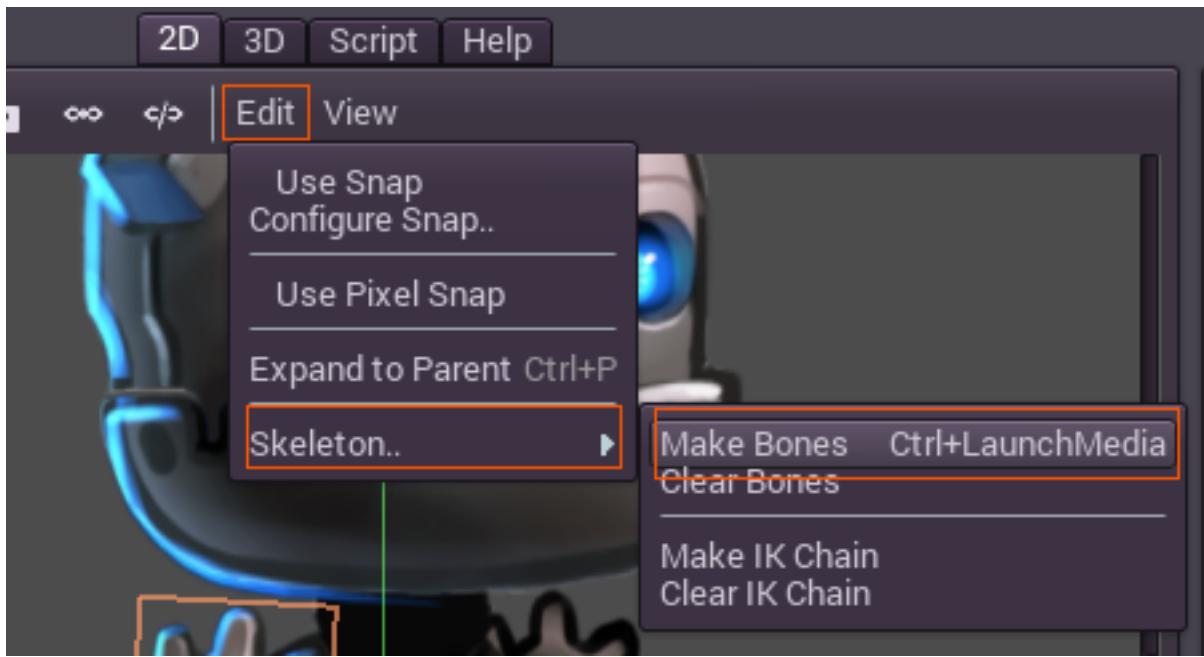
## Skeletons

Godot *does not really support actual skeletons*. What exists is a helper to create “bones” between nodes. This is enough for most cases, but the way it works is not completely obvious.

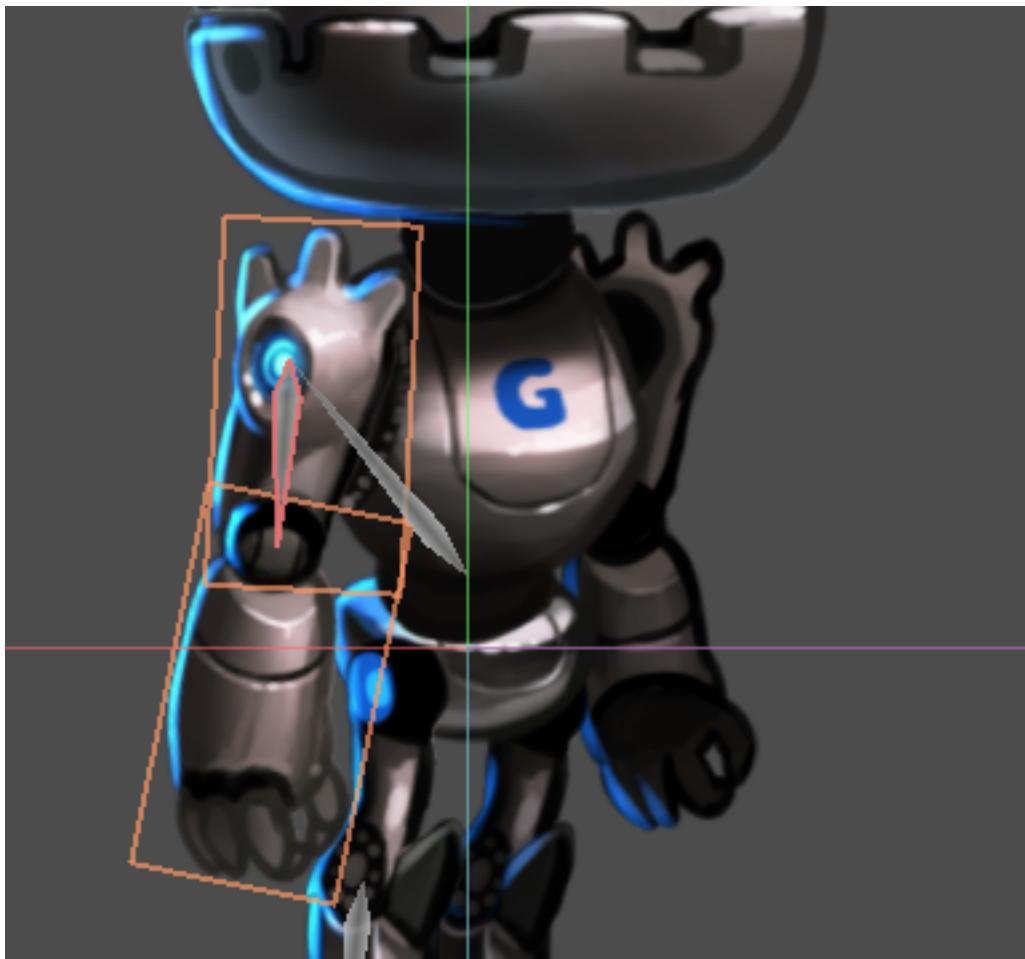
As an example, let’s turn the right arm into a skeleton. To create skeletons, a chain of nodes must be selected from top to bottom:



Then, the option to create a skeleton is located at Edit [STRIKEOUT:> Skeleton]> Make Bones:



This will add bones covering the arm, but the result is not quite what is expected.



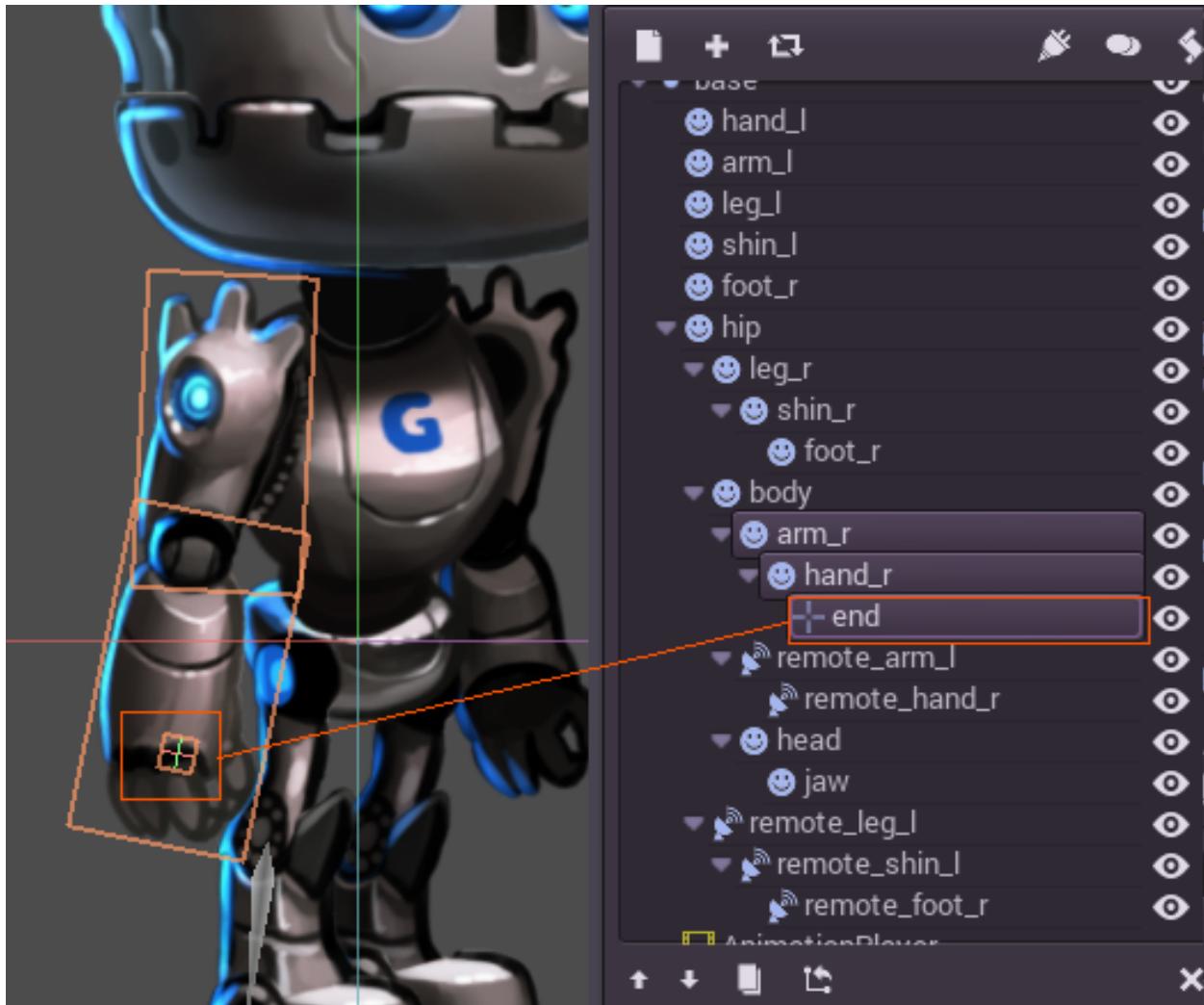
It looks like the bones are shifted up in the hierarchy. The hand connects to the arm, and the arm to the body. So the question is:

- Why does the hand lack a bone?
- Why does the arm connect to the body?

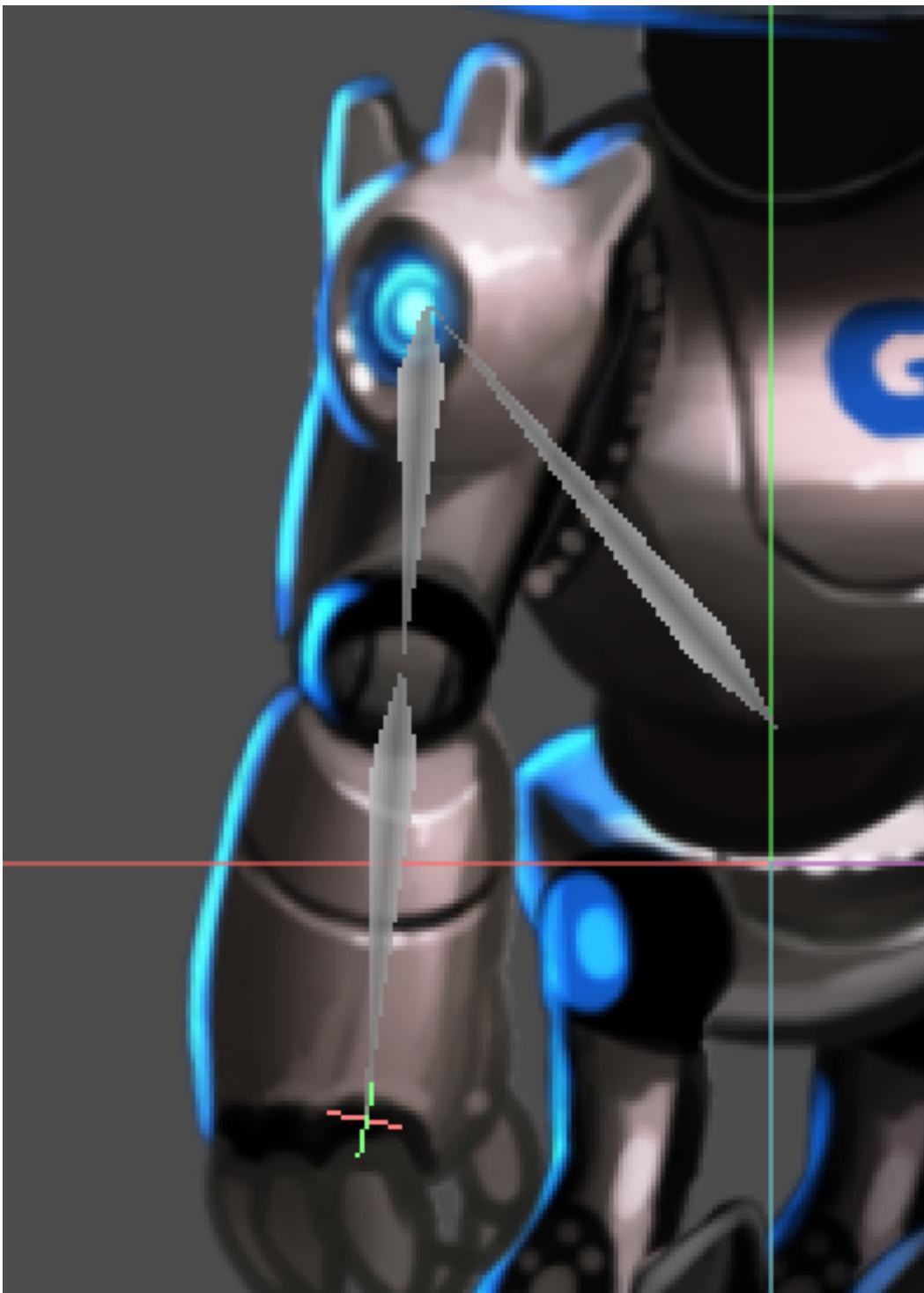
This might seem strange at first, but will make sense later on. In traditional skeleton systems, bones have a position, an orientation and a length. In Godot, bones are mostly helpers so they connect the current node with the parent. Because of this, **toggling a node as a bone will just connect it to 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

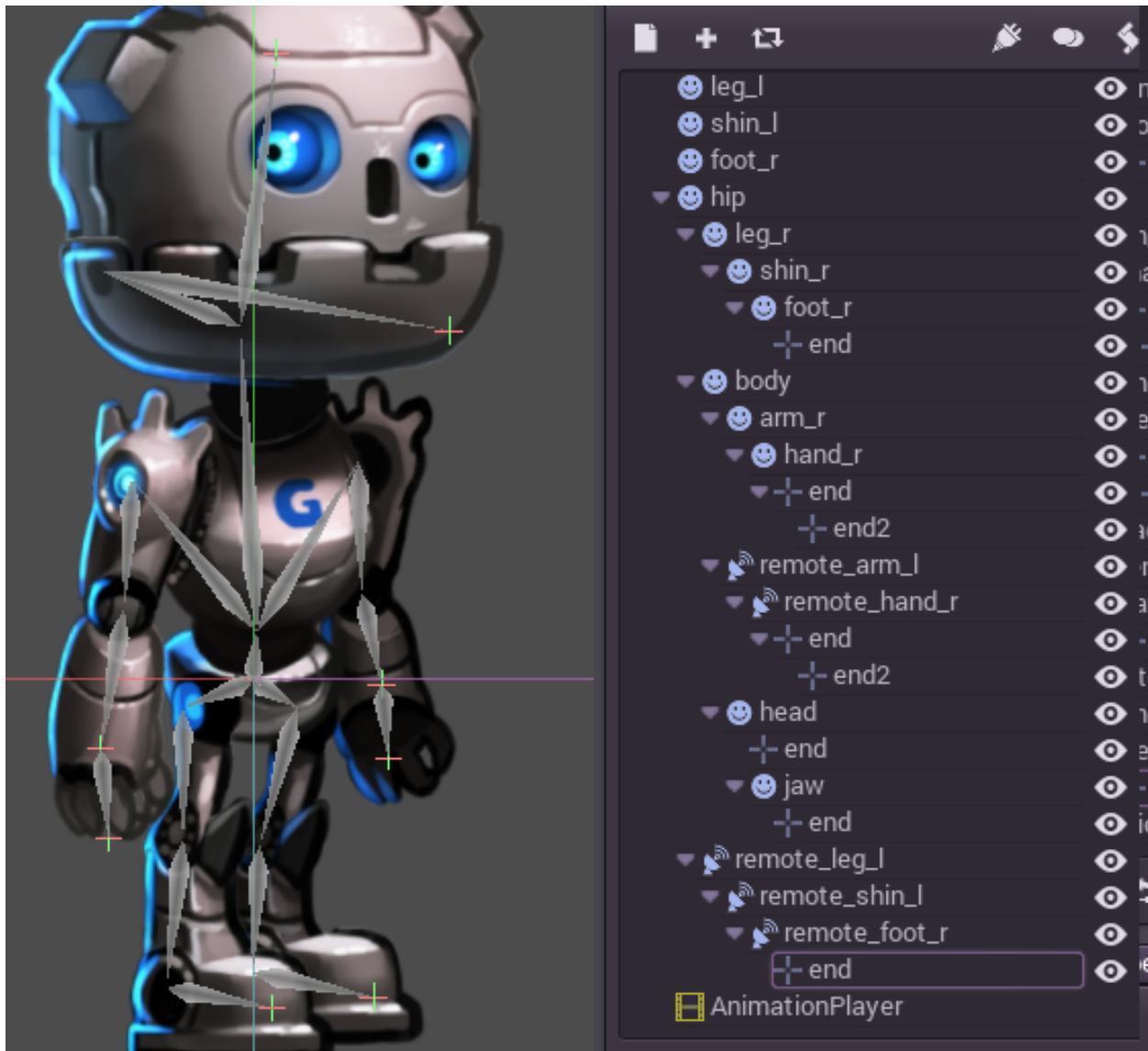


Now select the whole chain, from the endpoint to the arm and create bones:



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.

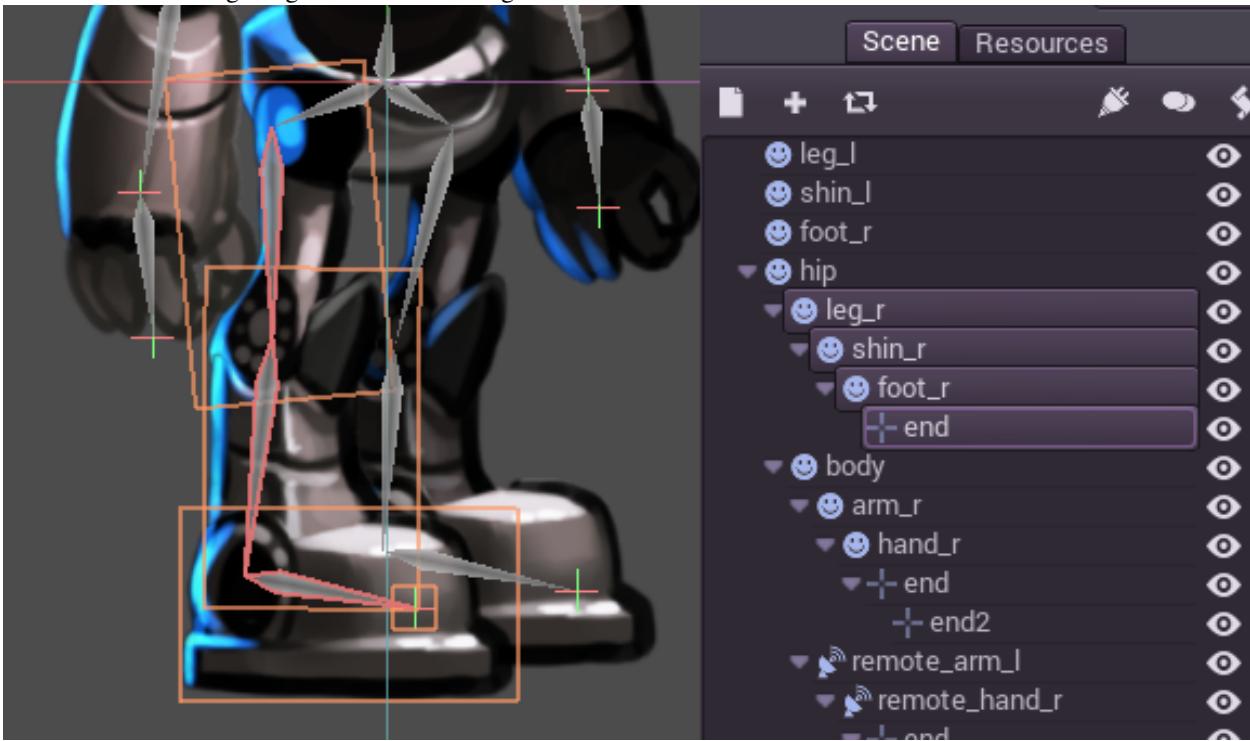
Now that a whole skeleton is rigged, the next step is setting up the IK chains. IK chains allow for more natural control of extremities.

## IK Chains

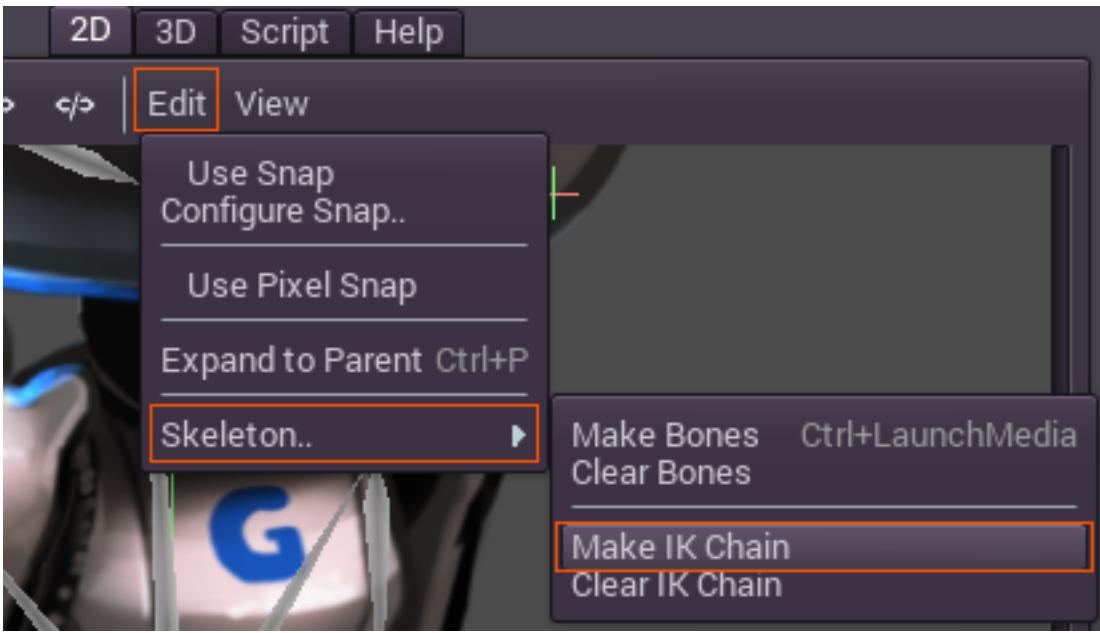
To add in animation, IK chains are a powerful tool. Imagine you want to pose a foot in a specific position in the ground. Moving the foot involves also moving the rest of the leg bones. Each motion of the foot involves rotating several other bones. This is quite complex and leads to imprecise results.

So, what if we could just move the foot and let the rest of the leg accommodate to the new foot position? This type of posing is called IK (Inverse Kinematic).

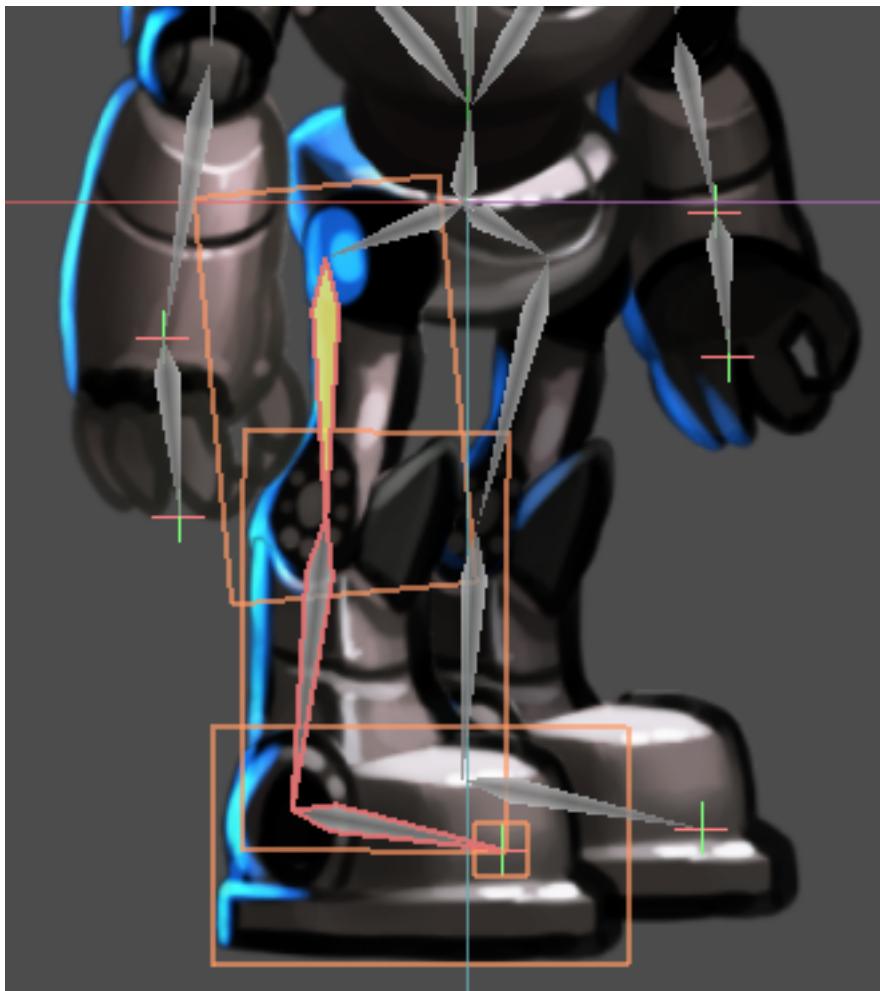
To create an IK chain, simply select a chain of bones from endpoint to the base for the chain. For example, to create an IK chain for the right leg select the following:



Then enable this chain for IK. Go to Edit [STRIKEOUT:> Skeleton]> Make IK Chain



As a result, the base of the chain will turn *Yellow*.



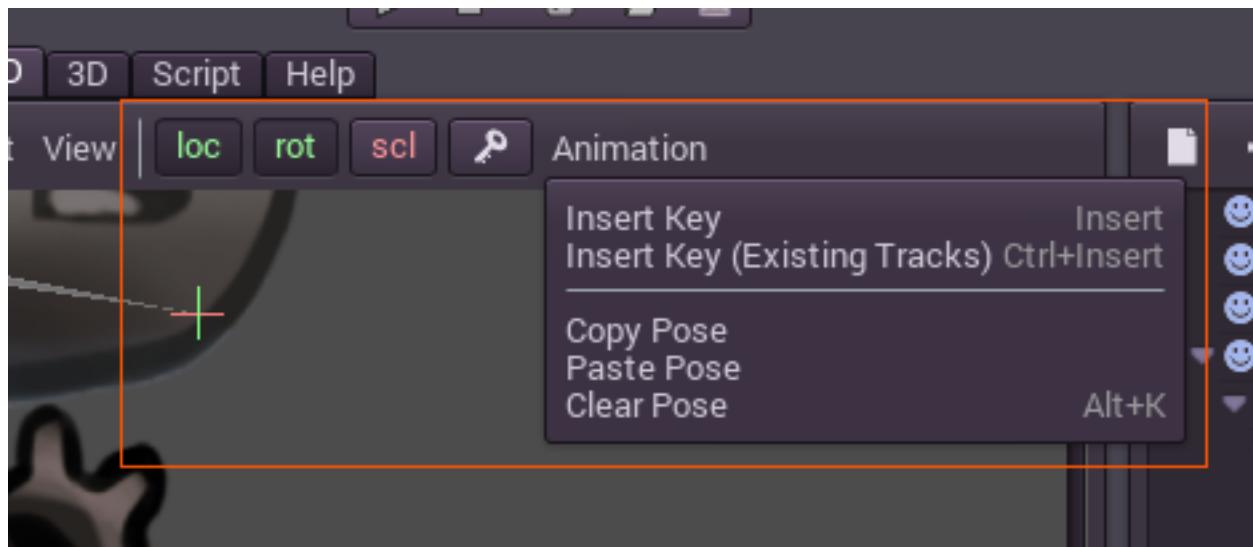
Once the IK chain is set-up, simply grab any of the bones in the extremity, any child or grand-child of the base of the chain and try to grab it and move it. Result will be pleasant, satisfaction warranted!

## 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 [[tutorial\_animation]].

### 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:



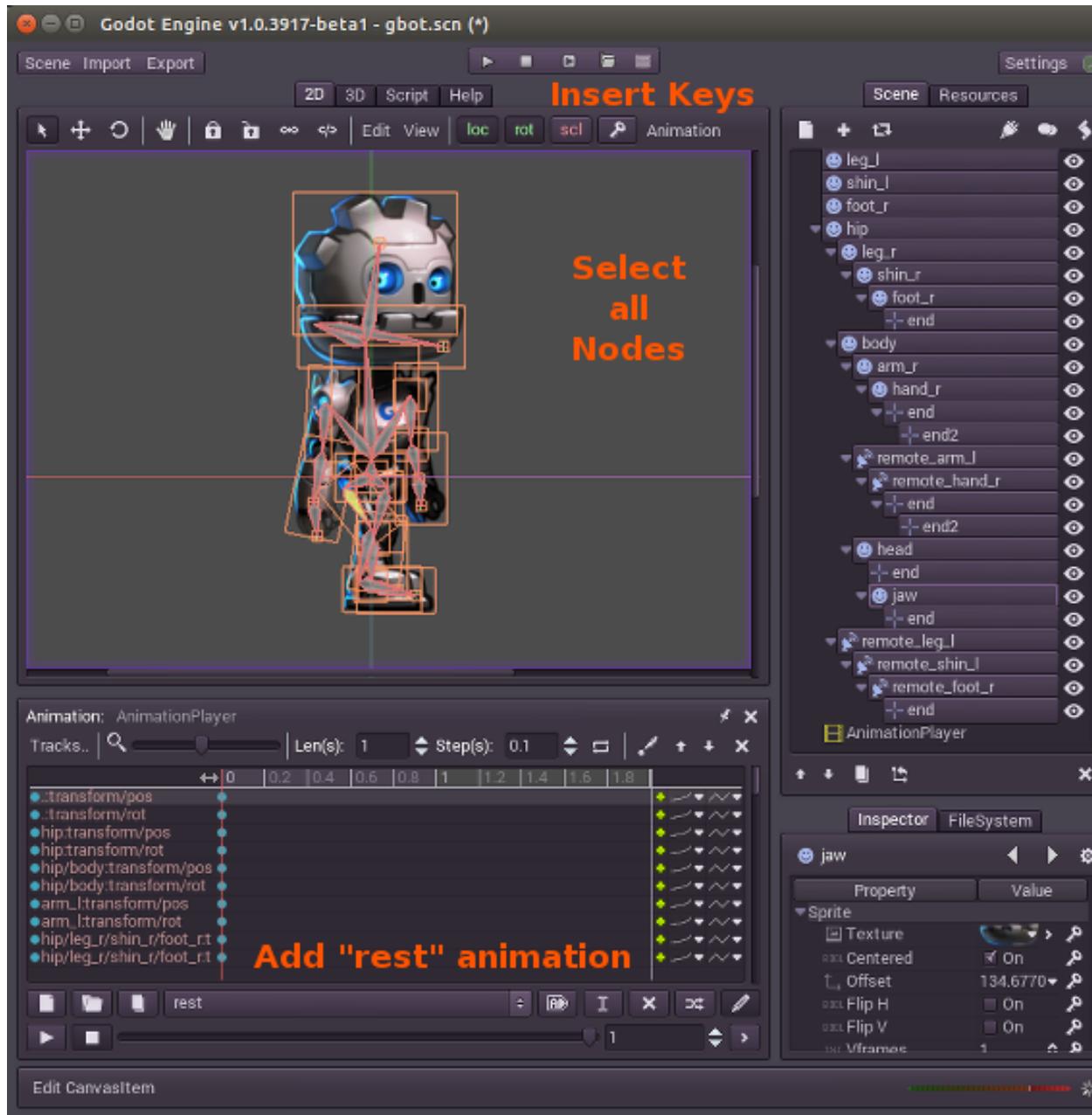
The key button will insert location/rotation/scale keyframes to the selected objects or bones. This depends on the mask enabled. Green items will insert keys while red ones will not, so modify the key insertion mask to your preference.

### 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:



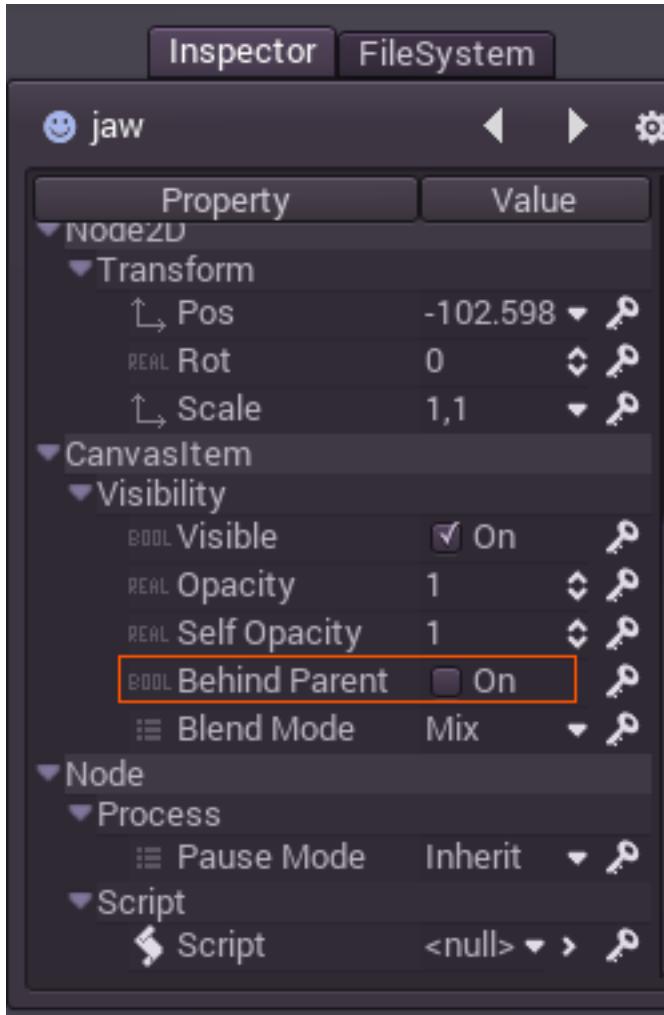
This will avoid the creation of extra animation tracks for the position that will remain unused.

## 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.

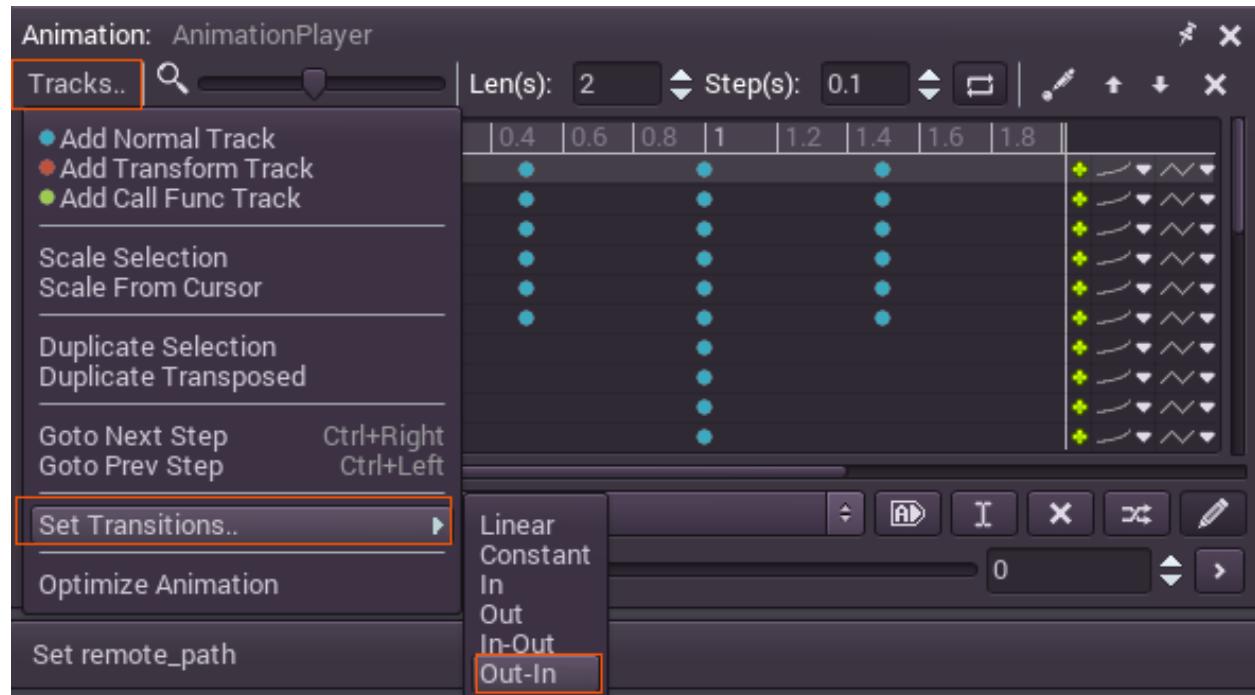
## Moving Sprites Above and Behind Others.

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:



### 3.1.7 Creating a Tilemap

### 3.1.8 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:



Collision can also be added to the tiles, allowing for both 2D side scroller or top down games.

### Making a Tileset

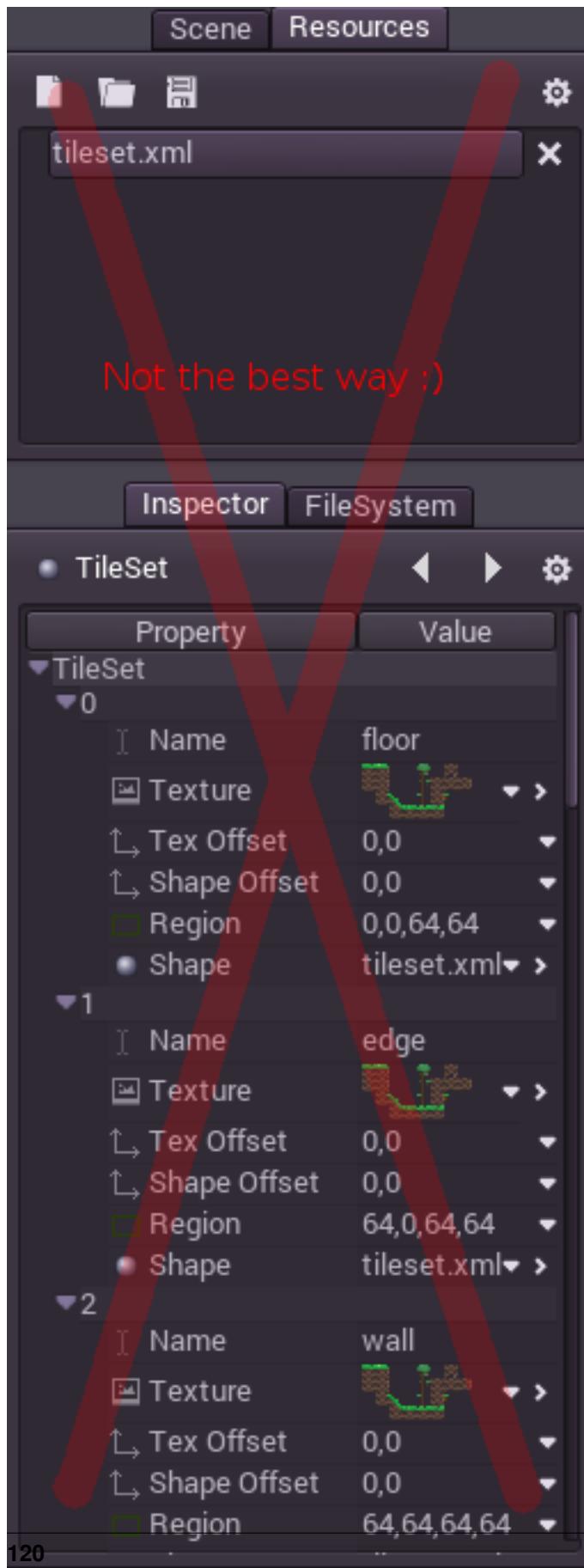
To begin with, a tileset needs to be made. Here are some tiles for it. They are all in the same image because artists will often prefer this. Having them as separate images also works too.



Create a new project and throw the above png image inside.

### Create the TileSet Scene

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:



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 with the export tool!

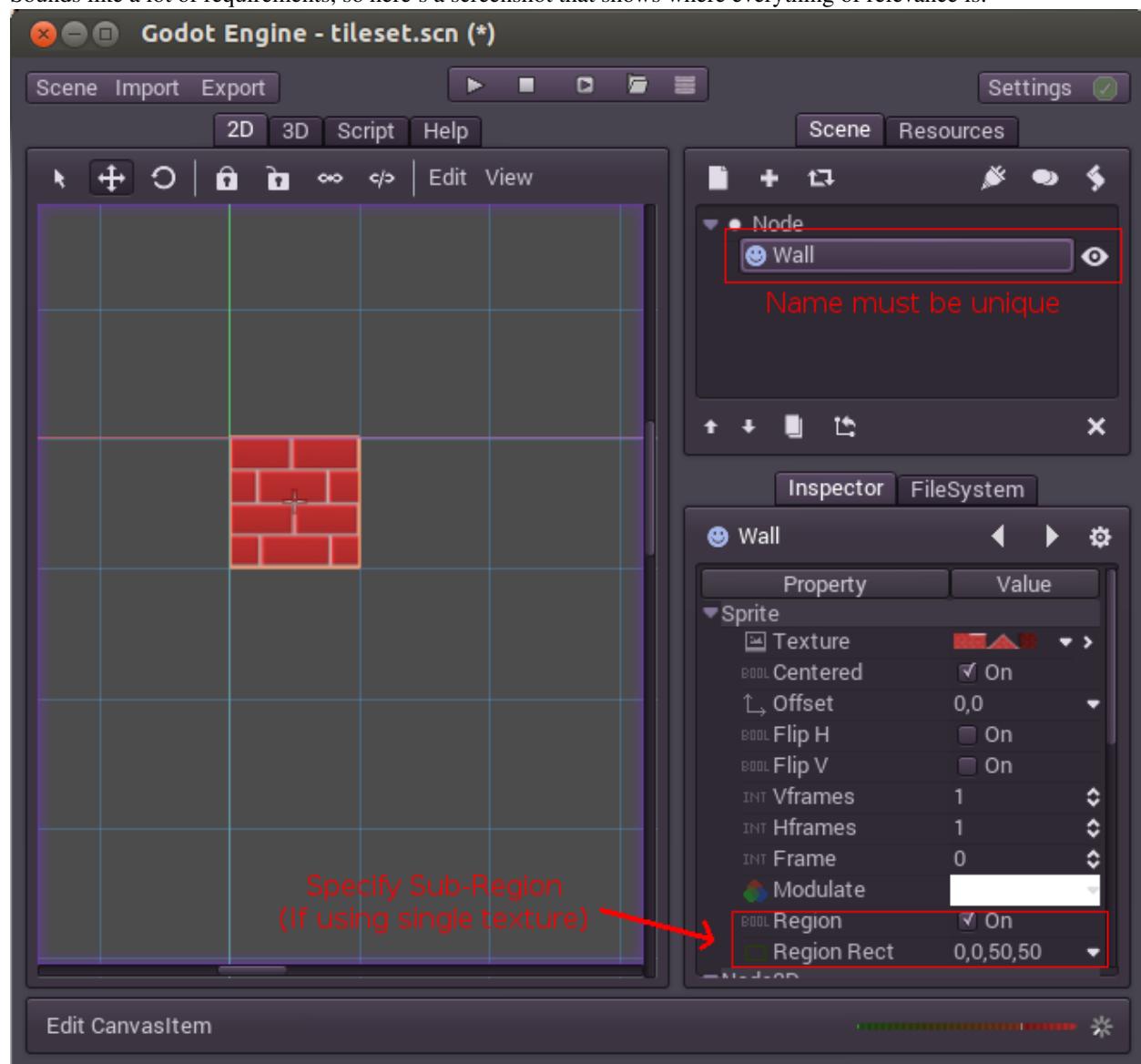
## TileSet Scene

Create a new scene with a regular node or node2d as root. For each new a sprite will be added. 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 of the sprites is the source texture is being edited.

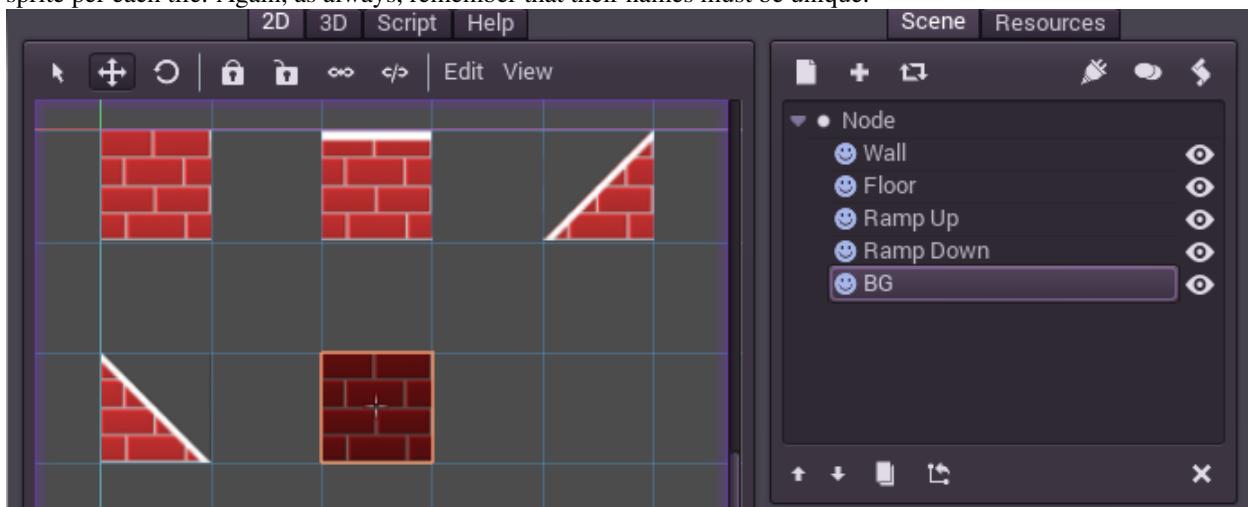
Finally, make sure to name your sprite node correctly, this will ensure that, in subsequent edits to the tileset (for example, if added collision, changed the region, etc), the tile will still be **identified correctly and updated**. This name should be unique.

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



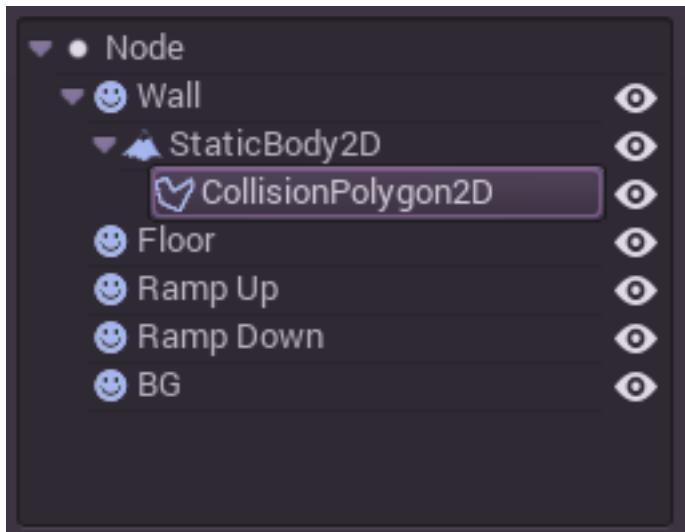
Continue adding all the tiles, adjust the offsets if needed (if you use multiple tiles in a single image) unless there is a

sprite per each tile. Again, as always, remember that their names must be unique.

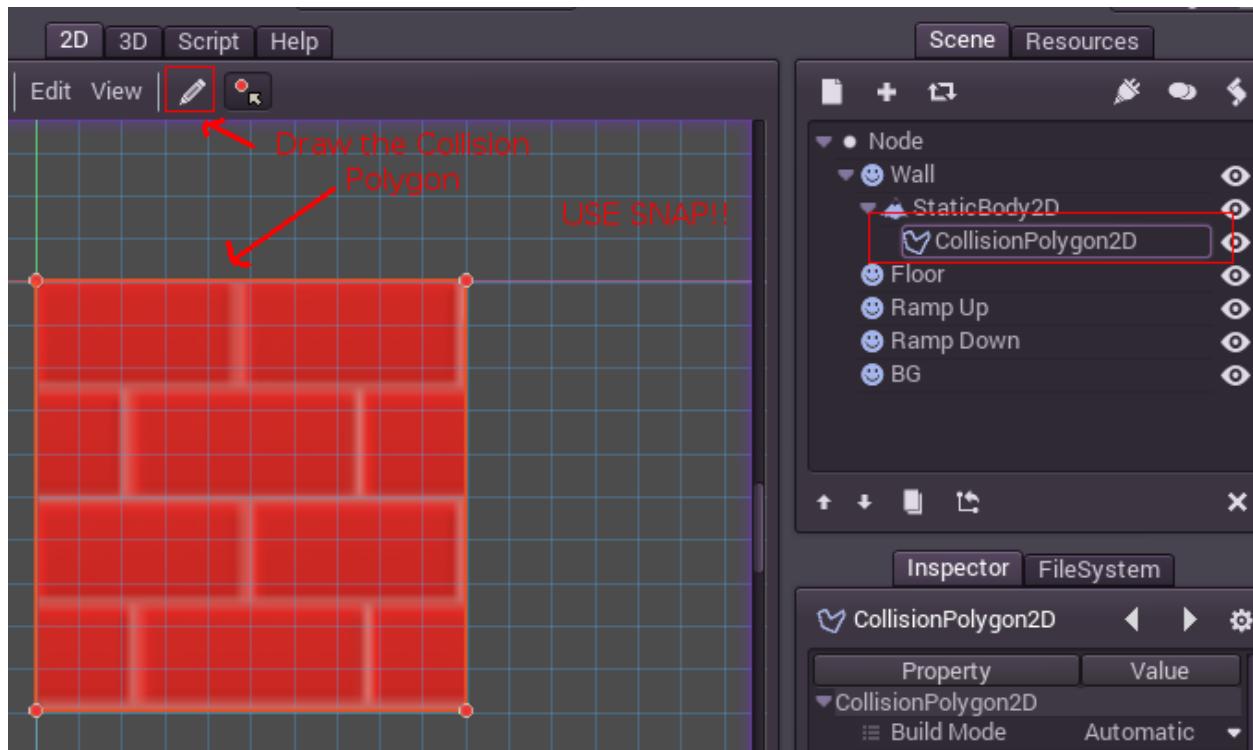


## Collision

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 `CollisionPolygon2D`. The later is recommended because it's 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.



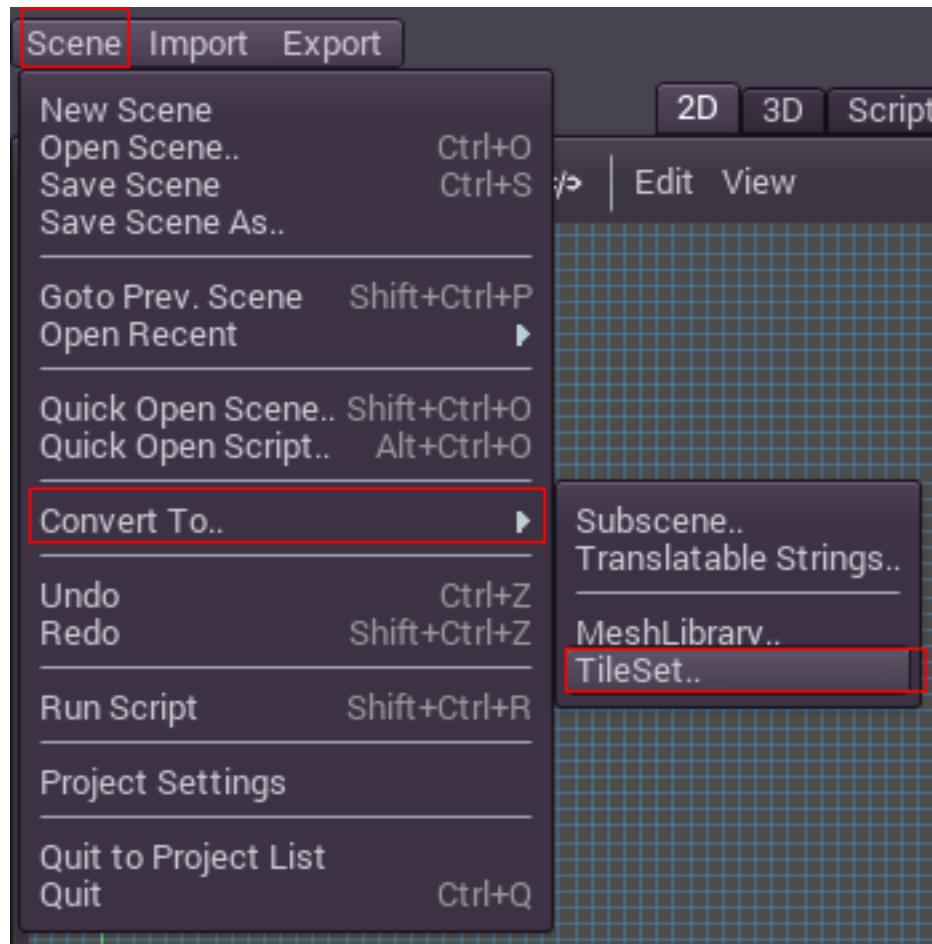
Keep adding collisions to tiles until we are done. Note that BG is just a BG so we don't care about it.



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:

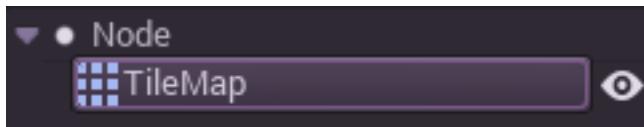


Then choose a filename, like “mytiles.res”. Make sure the “Merge With Existing” option is toggled on. This way, every time the tileset resource file is overwritten, existing tiles are merged and updated (they are referenced by their unique name, so again, **name your tiles properly**).

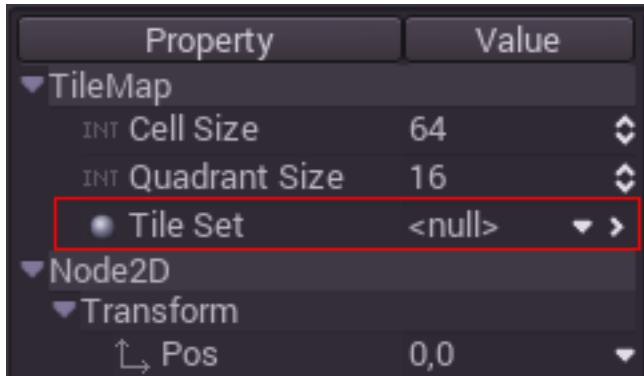


## Using the TileSet in a TileMap

Create a new scene, use any node or node2d as root, then create a TileMap as a child.



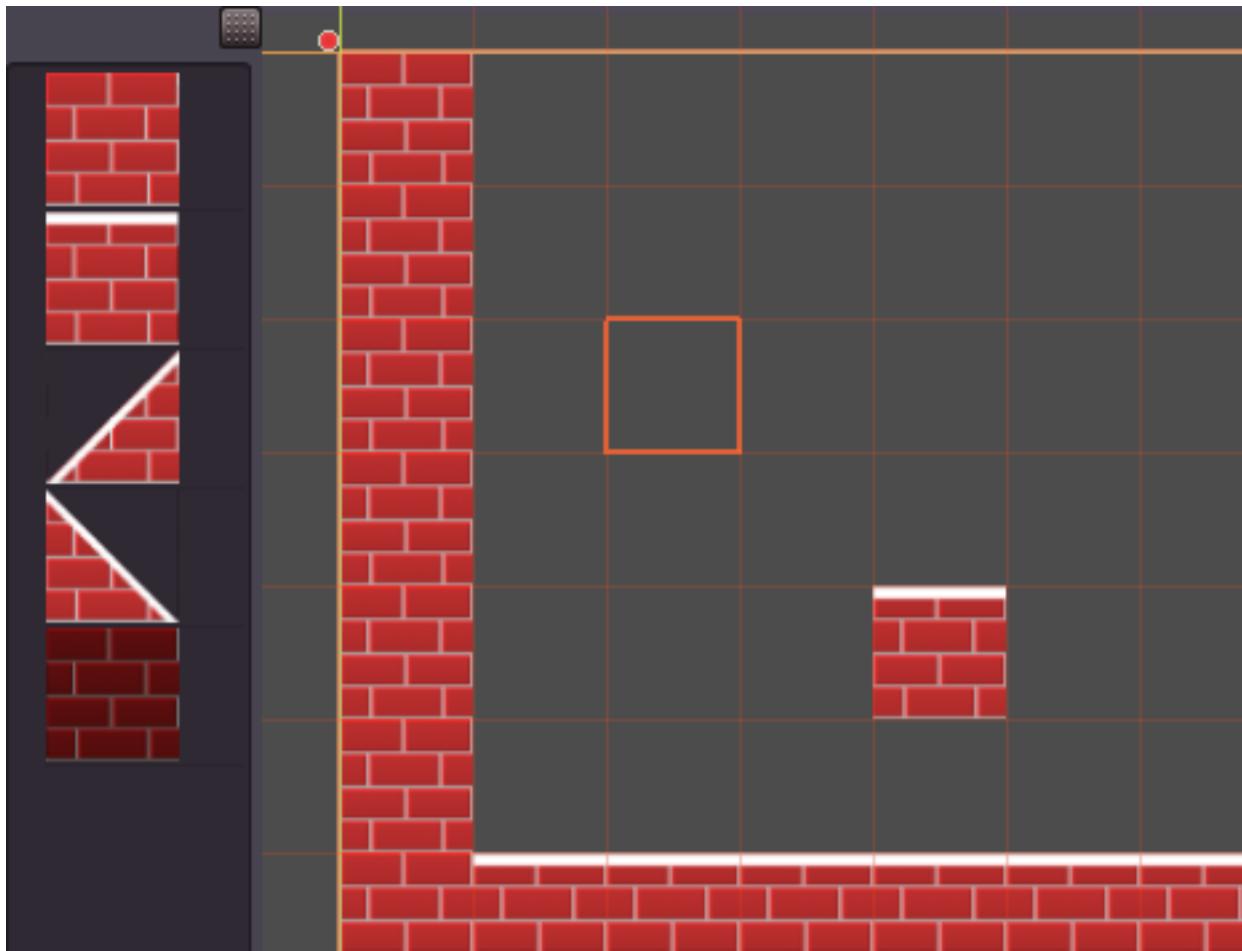
Go to the tileset property of this node and assign the one created in previous steps:



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.

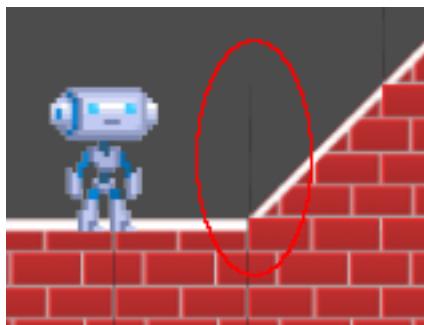


To avoid moving and selecting the tilemap node accidentally (something common given it's a huge node), it is recommended that you lock it, using the lock button:



### 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 first try the options below first.
- Disable filtering for either the tileset texture or the entire image loader (see the [[Image Files]] asset pipeline tutorial).
- Enable pixel snap (Set: Scene [STRIKEOUT:> Project Settings]> rasterizer/uxe\_pixel\_snap” to true).
- Viewport Scaling can often help shrinking the map (see the [[Viewports]] tutorial).

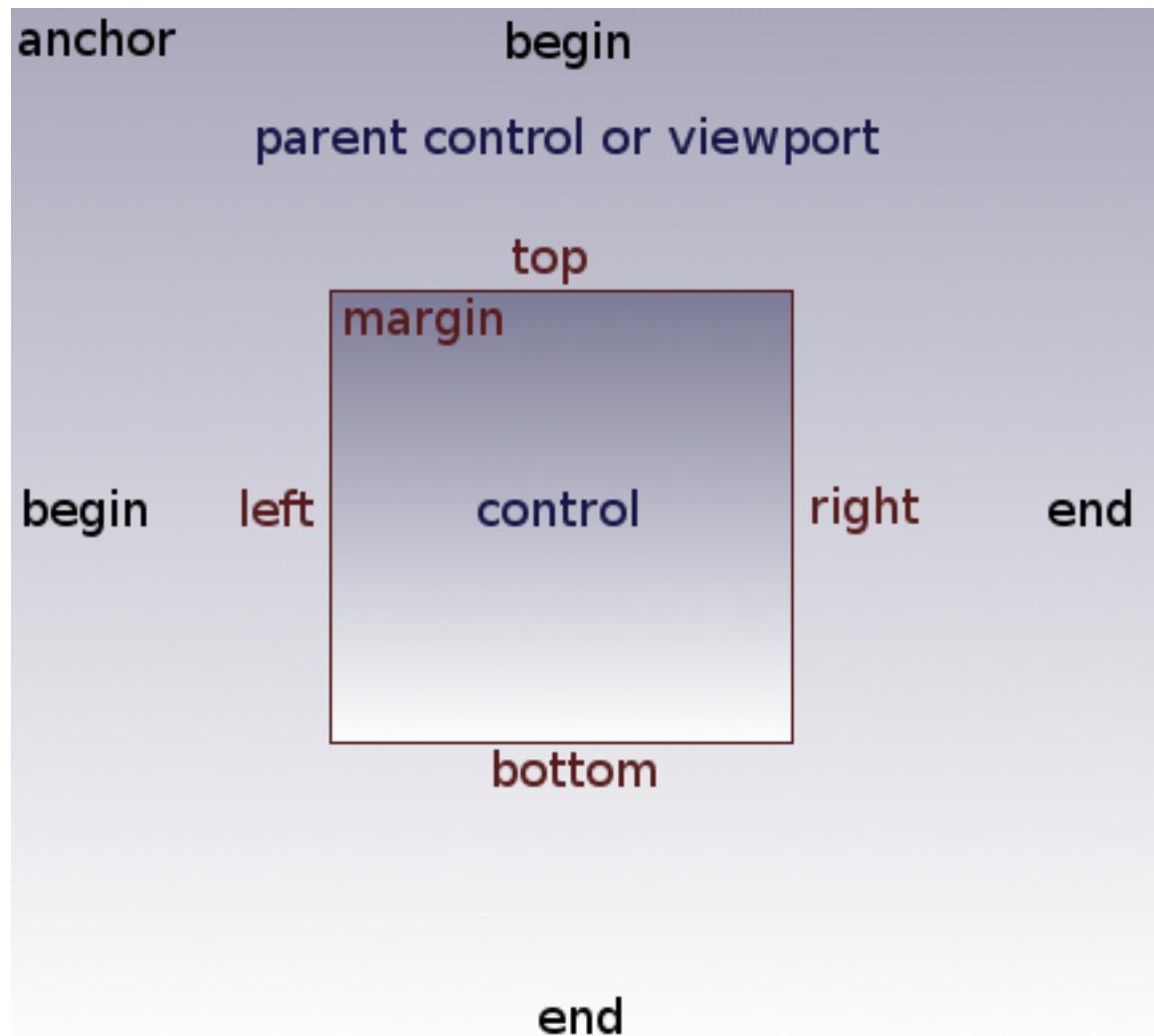
## 3.2 Graphical user interface (GUI)

### 3.2.1 Size and Anchors

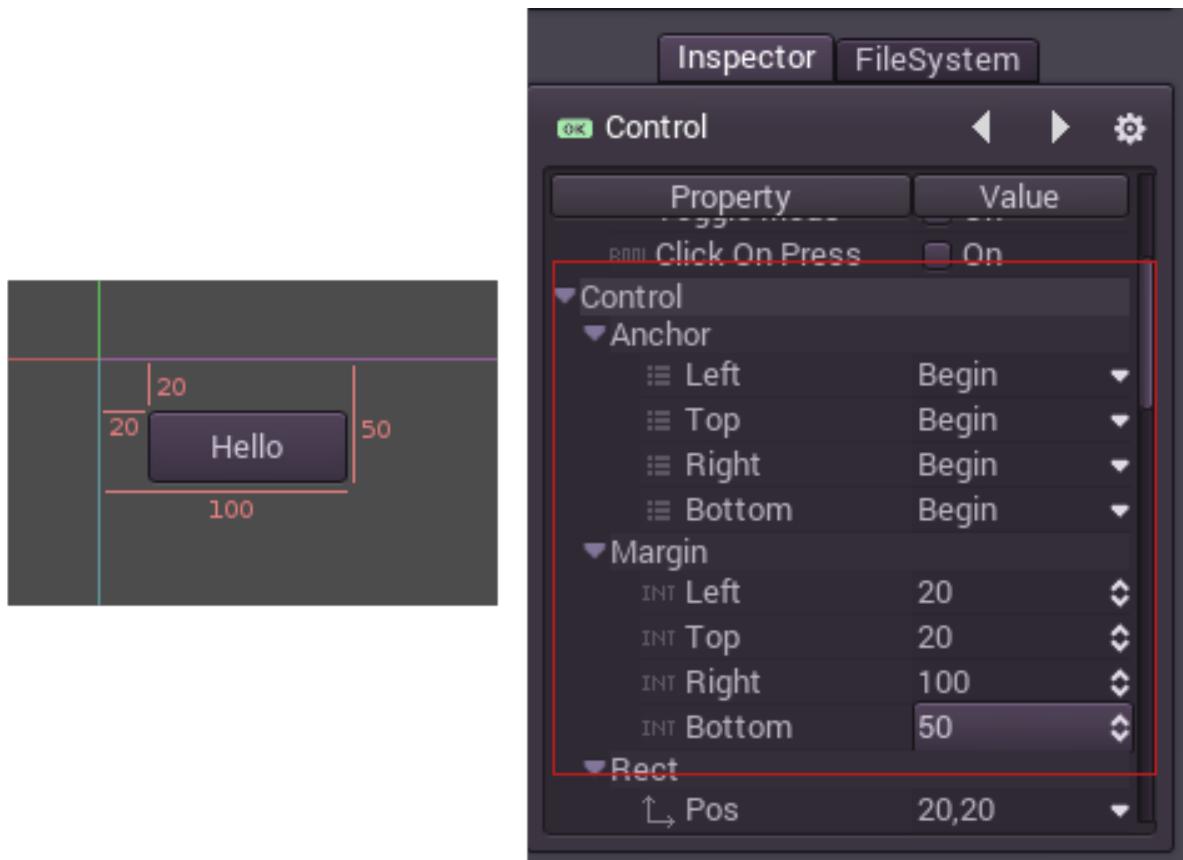
If a game was to be always run in the same device and at the same resolution, positioning controls would be a simple matter of setting the position and size of each one of them. Unfortunately, it is rarely the case.

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.

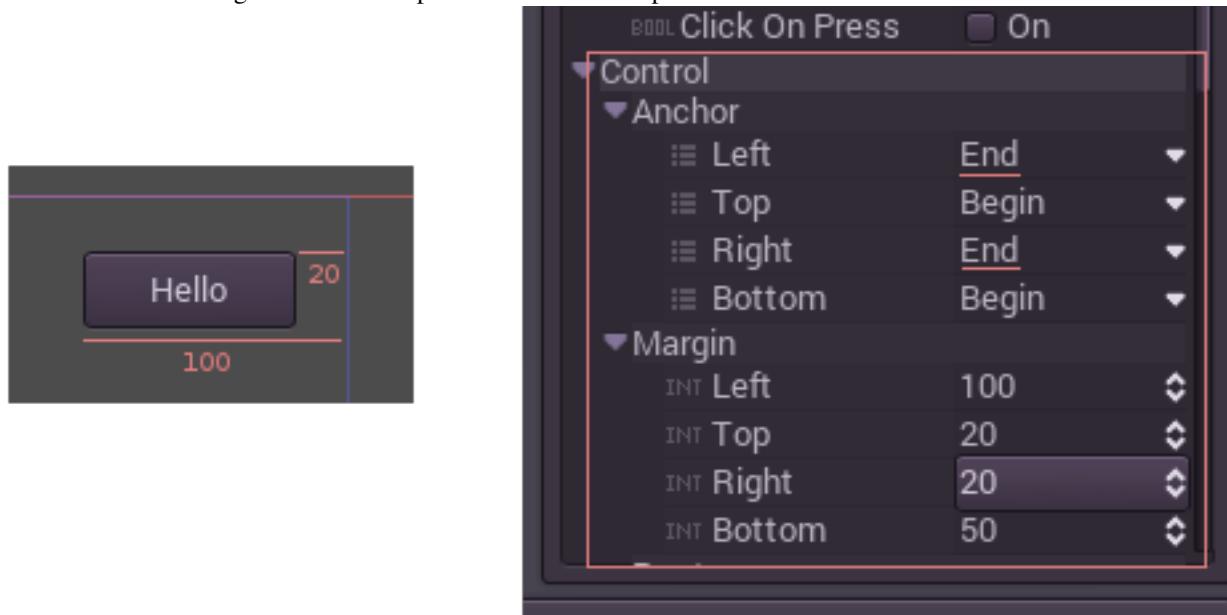
There are several ways to handle this, but for now let's just imagine that the screen resolution has changed and the controls need to be re-positioned. Some will need to follow the bottom of the screen, others the top of the screen, or maybe the right or left margins.



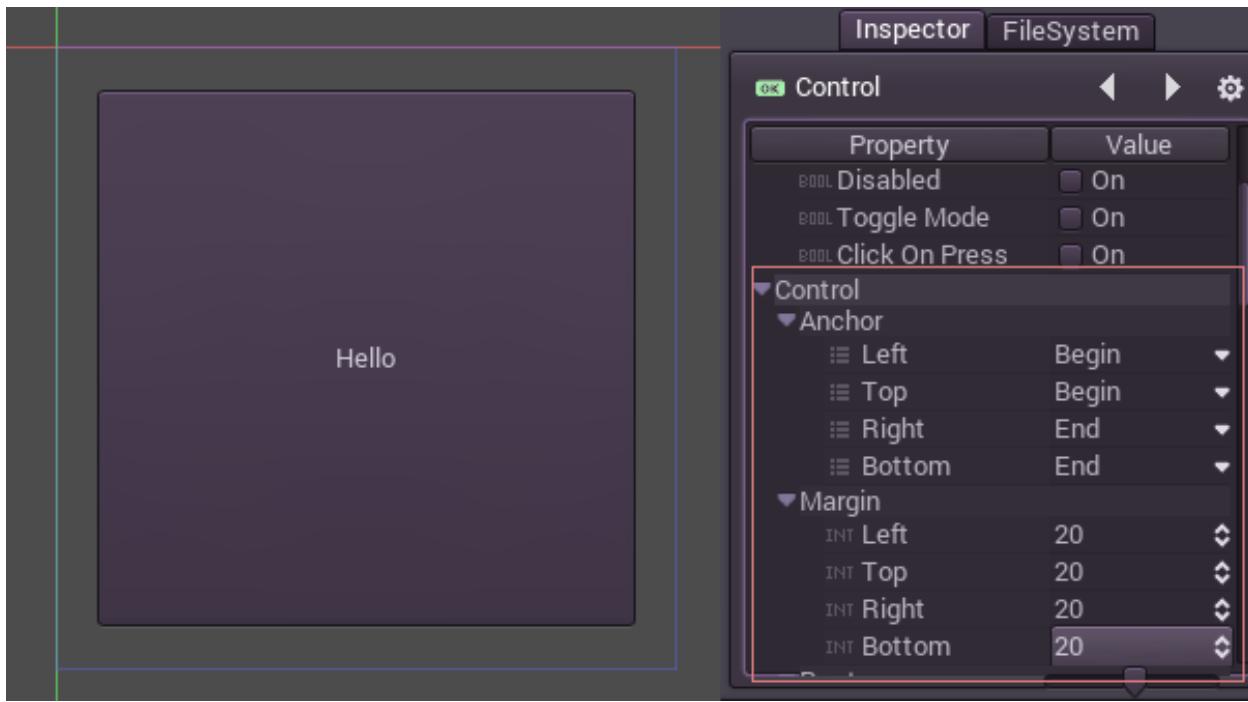
This is done by editing the *margin* properties of controls. Each control has four margins: left, right, bottom and top. By default all of them represent a distance in pixels relative to the top-left corner of the parent control or (in case there is no parent control) the viewport.



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 it's 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.

### 3.2.2 Skinning a GUI

#### 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 (specially 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:

```
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.

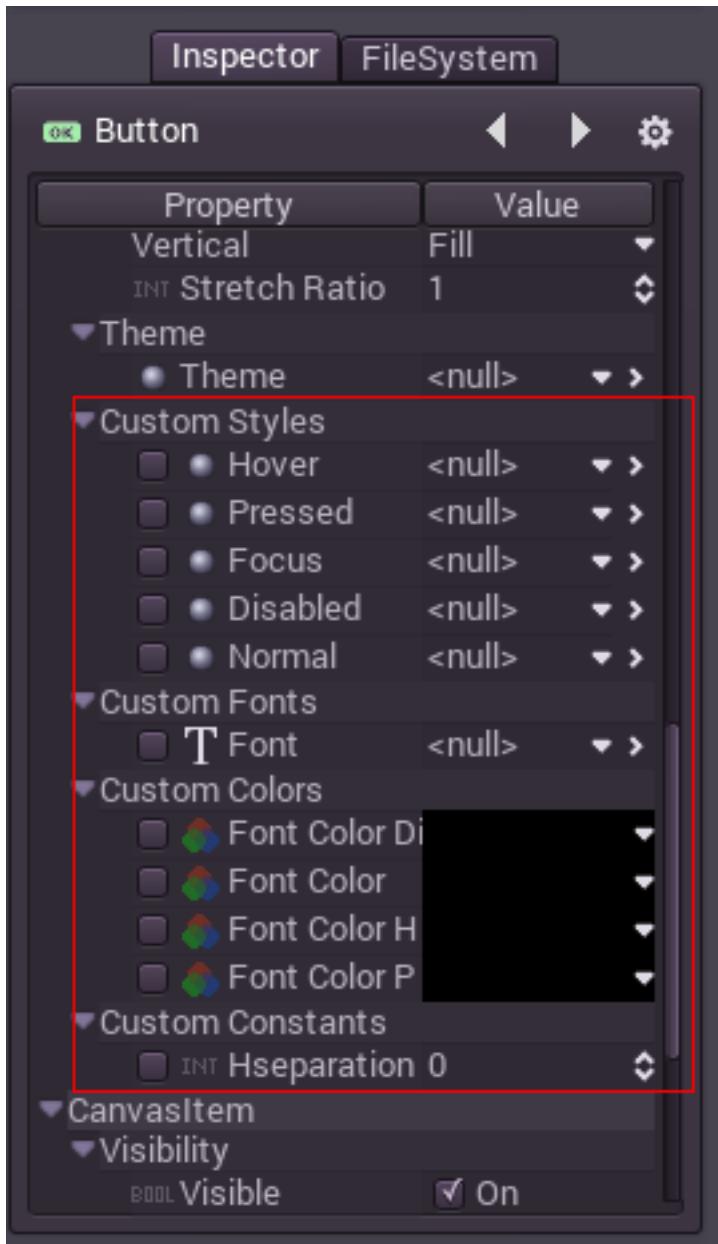
It is possible to override those options without using the theme directly and only for a specific control by using the [override API in Control](#):

```
var l = Label.new()
l.add_color_override("font_color", Color(1.0, 1.0, 1.0))
```

In the inline help of Godot (help tab) you can check which theme options are overrideable. This is not yet available in the wiki class reference, but will be soon.

## Customizing a Control

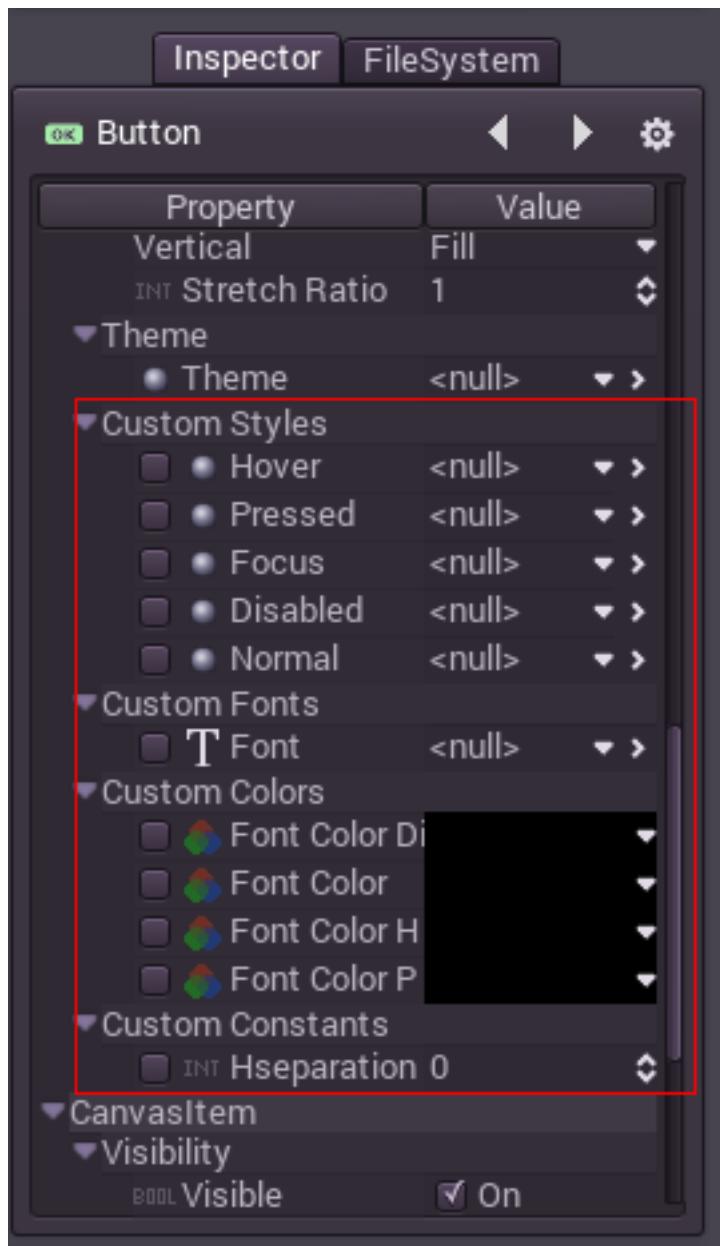
If only a few controls need to be skinned. It is often not necessary to create a new theme. Controls offer their theme options as special kind of properties. If checked, overriding will take place:



As can be seen 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

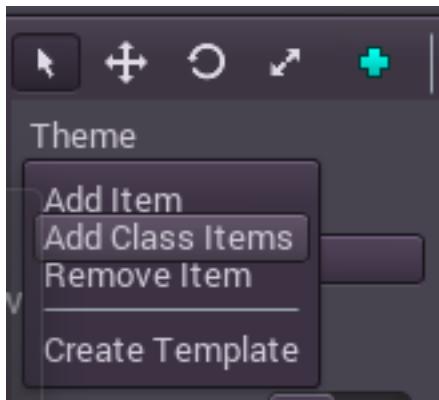
The simplest way to create a theme is to edit a theme resource. Create a Theme from the resource menu, the editor will appear immediately. Following to this, save it (to, as example, mytheme.thm):



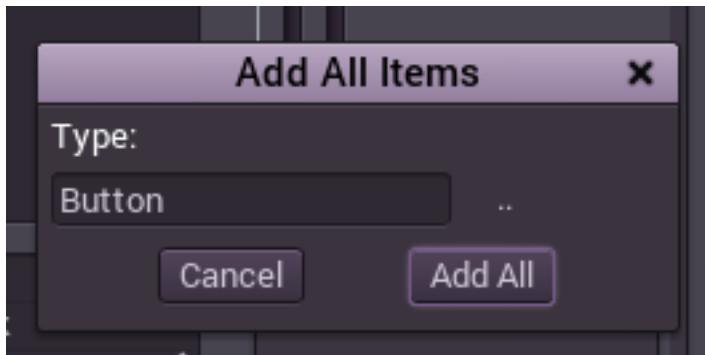
This will create an empty theme that can later be loaded and assigned to controls.

### Example: Themeing a Button

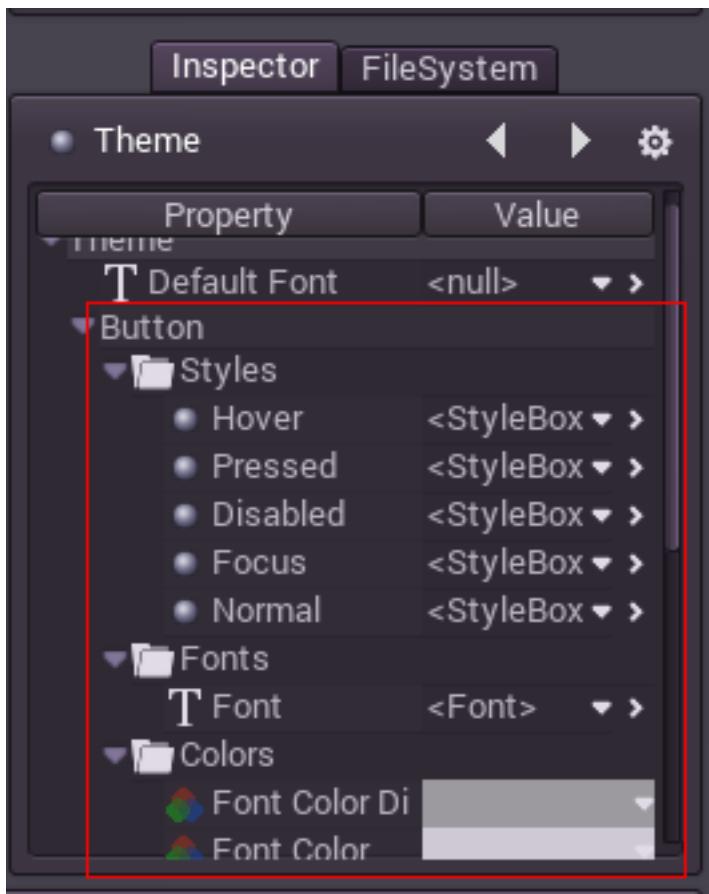
Take some assets attachment:skin\_assets.zip, go to the “theme” menu and select “Add Class Item”:



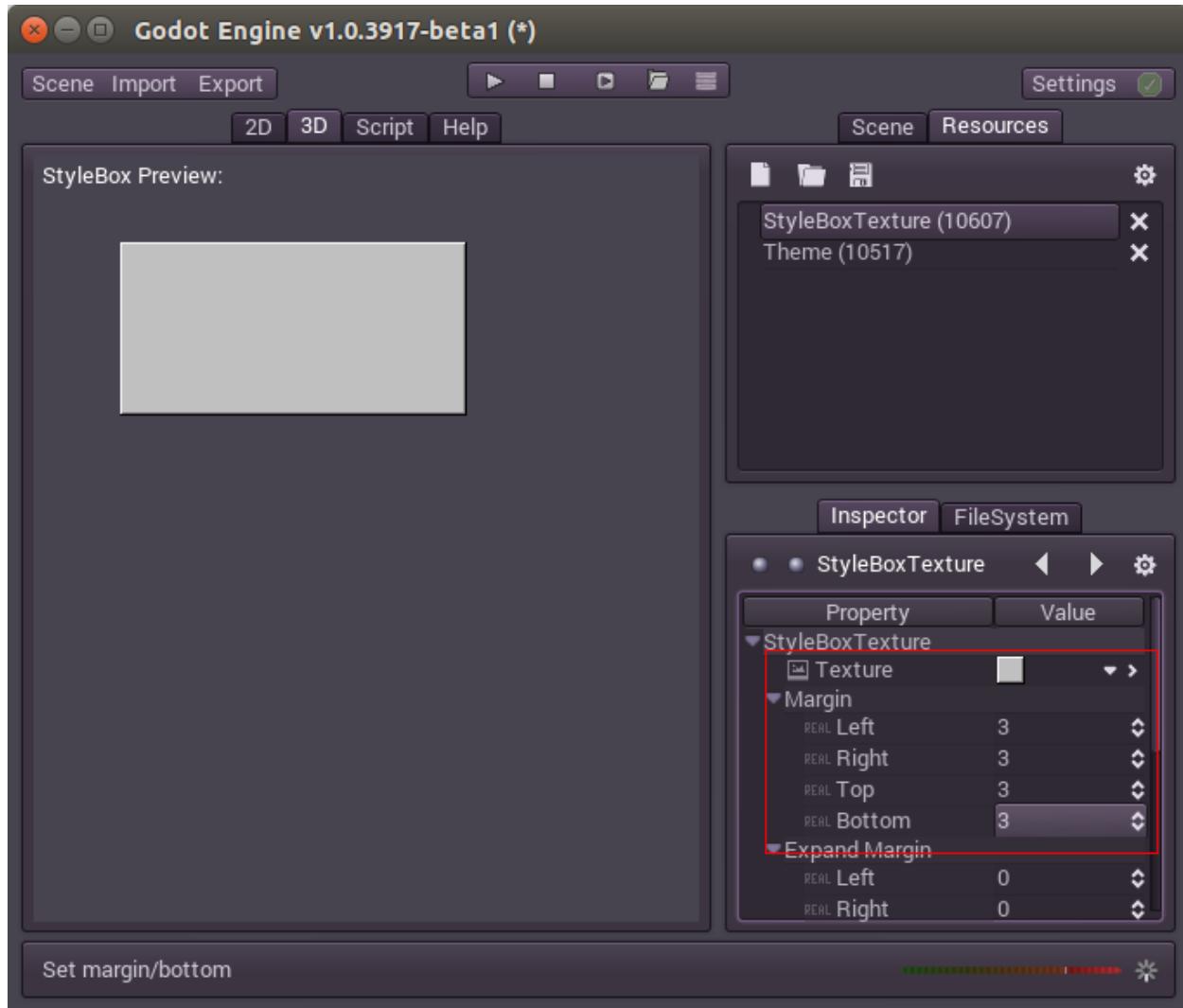
A menu will appear prompting the type of control to create. Select “Button”:



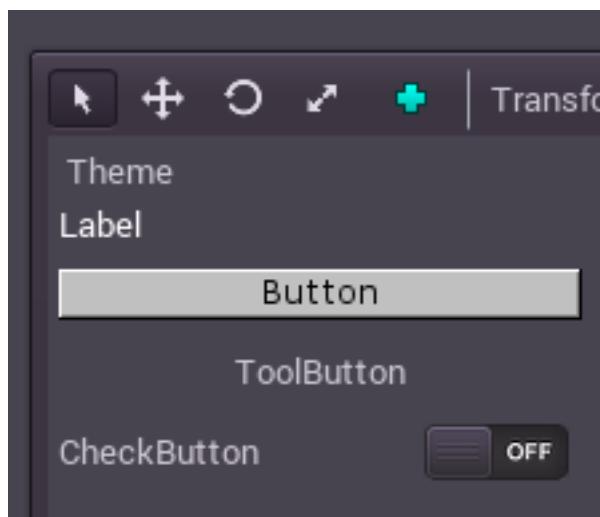
Immediately, all button theme options will appear in the property editor, where they can be edited:



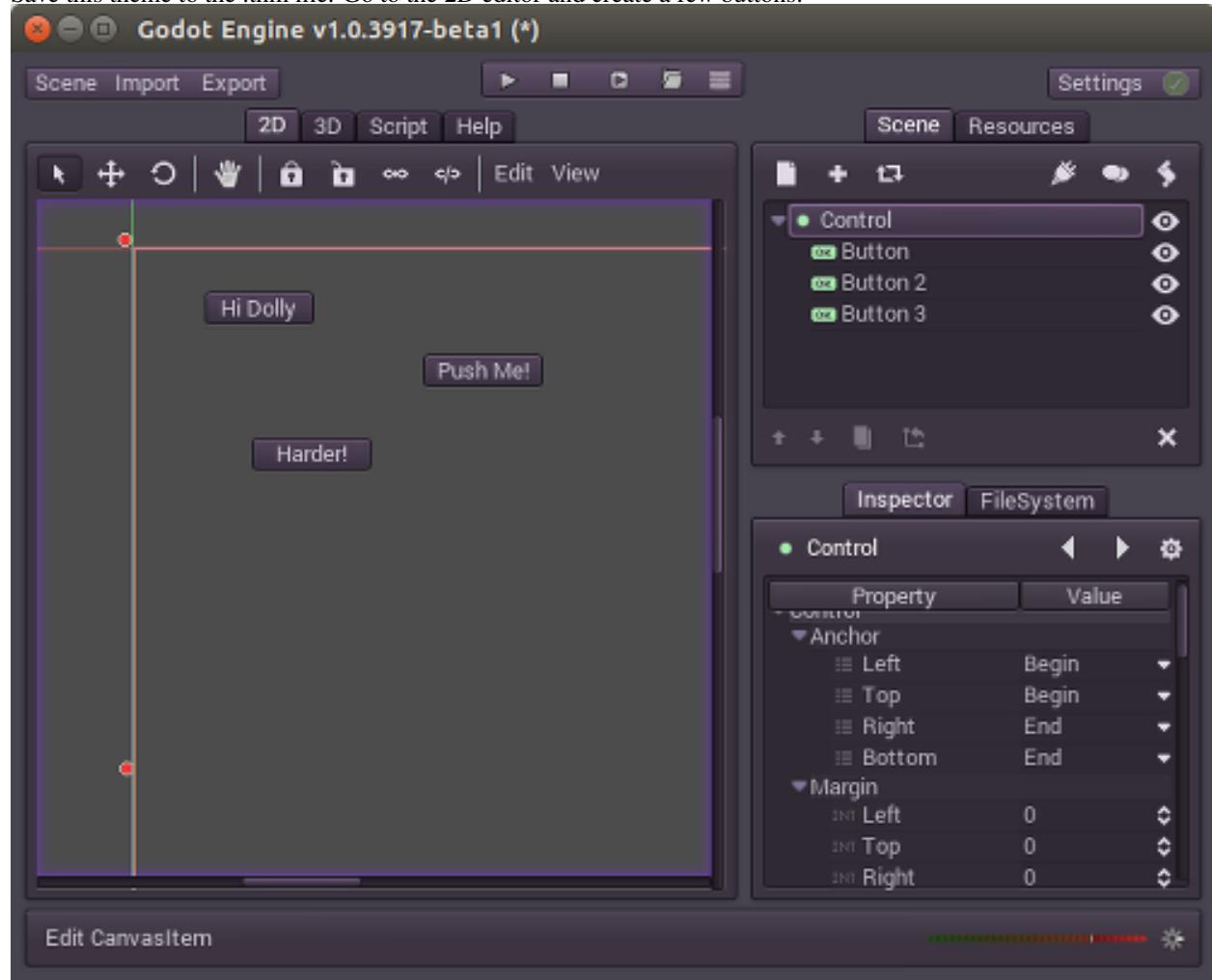
Select the “normal” stylebox and create a new “StyleBoxTexture”, then edit it. A texture stylebox basically contains a texture and the size of the margins that will not stretch when the texture is stretched. This is called “3x3” stretching:



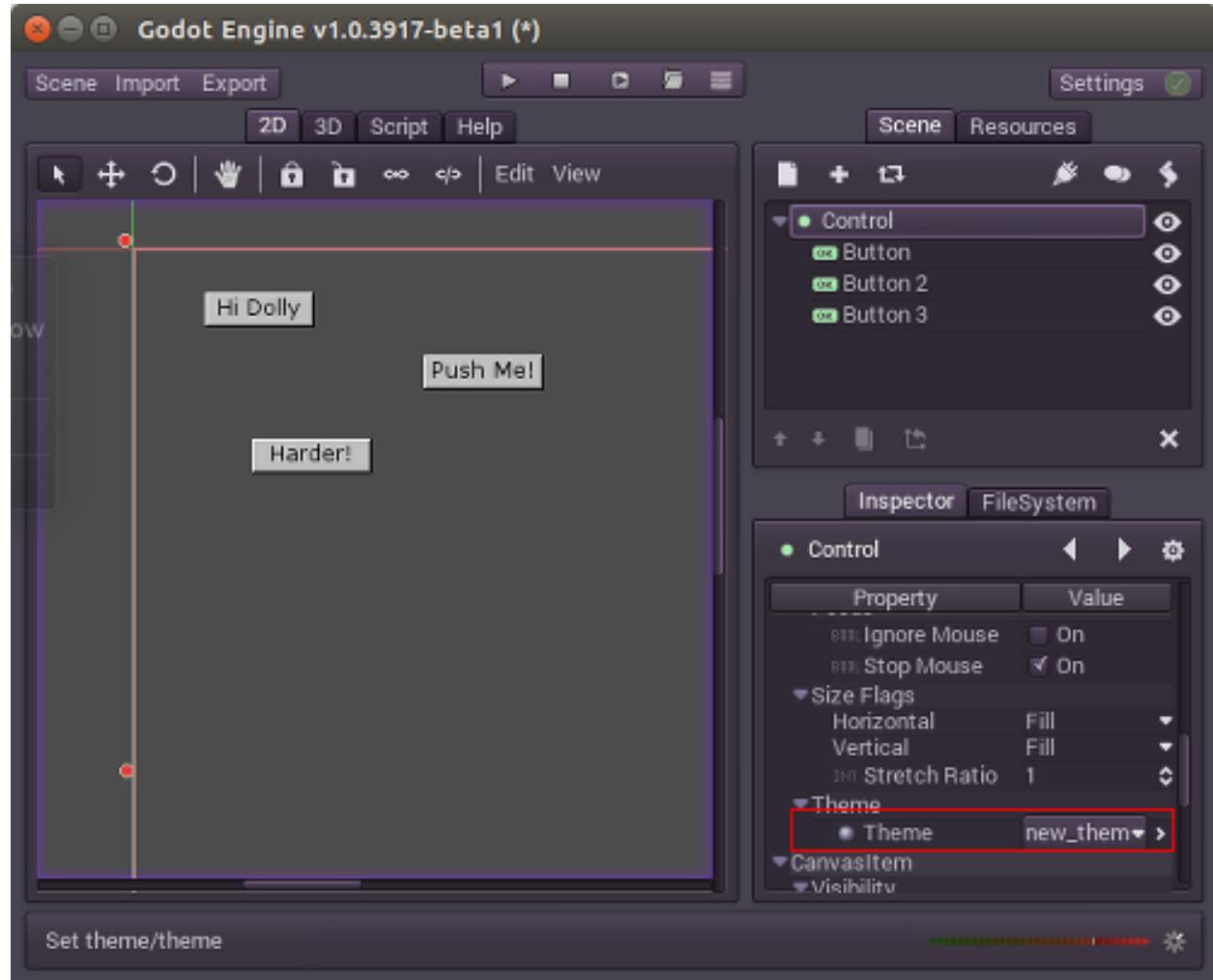
Repeat the steps and add the other assets. There is no hover or disabled image in the example files, so use the same stylebox as in normal. Set the supplied font as the button font and change the font color to black. Soon, your button will look different and retro:



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!

### 3.2.3 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 Draw 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 joypad 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

```
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:

```
func get_minimum_size():
    return Vector2(30, 30)
```

Or alternatively, set it via function:

```
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 left button was pressed over this control (control always captures input until button is released)
- Control provides keyboard/joypad focus via [Control.set\\_focus\\_mode](#).

This function is [Control.\\_input\\_event\(event\)](#). 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 [[Input Events]] 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_THEME_CHANGED):
        pass # theme used to draw the control changed
        # update and redraw is recommended if using a theme
    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
```

## 3.3 Physics

### 3.3.1 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:

```
#create a circle
var c = CircleShape2D.new()
c.set_radius(20)

#create a box
var b = RectangleShape2D.new()
b.set_extents(Vector2(20,10))
```

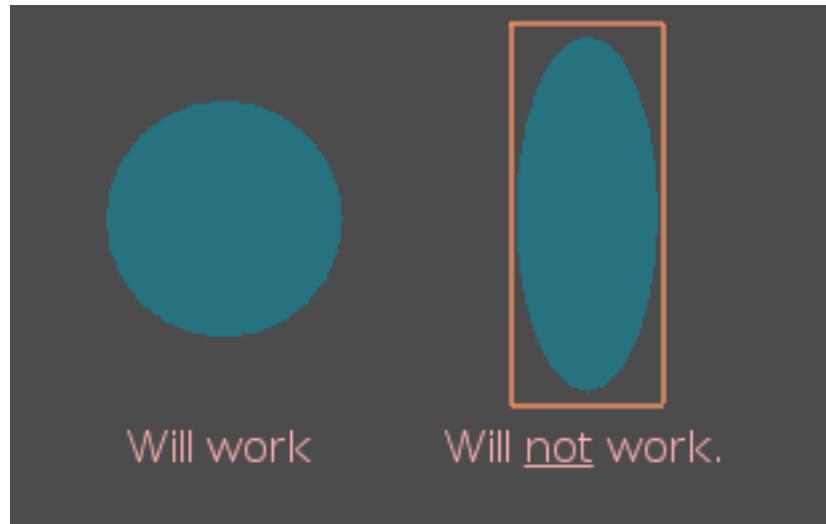
The main use for shapes is checking collision/intersection and getting resolution information. Shapes are mostly convex, (except the concavepolygon one, which is just a list of segments to check collision against). This collision check is done easily with the built-in functions like:

```
#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 can check collision while 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!



## But 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:

```
for i in colliders:  
    for j in colliders:  
        if (i.collides(j)):  
            do_collision_code()
```

But this scales really bad. Let's imagine there are only 100 objects in the game. This means that  $100 \times 100 = 10000$  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.

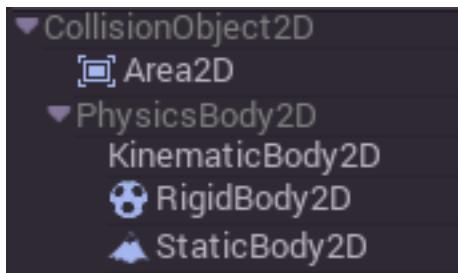
### Units of Measure

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

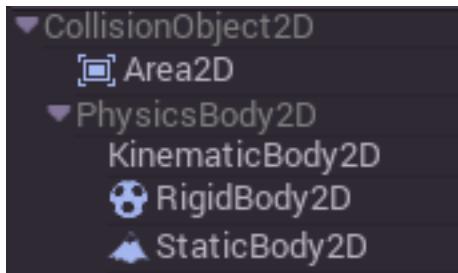
transform. This means that all collisionable objects in Godot can use multiple shapes at different transforms (offset/scale/rotation). Just remember that, as mentioned before, **non-uniform scale will not work for circle and capsule shapes**.



### StaticBody2D

The simplest node in the physics engine is the **StaticBody2D**, which provides a static collision. This means that other objects can collide against it, but **StaticBody2D** will not move by itself or generate any kind of interaction when colliding other bodies. It's just there to be collided.

Creating one of those bodies is not enough, because it lacks collision:



From the previous point, we know that **CollisionObject2D** derived nodes have an internal lists of shapes and transforms for collisions, but how to edit them? There are two special nodes for that.

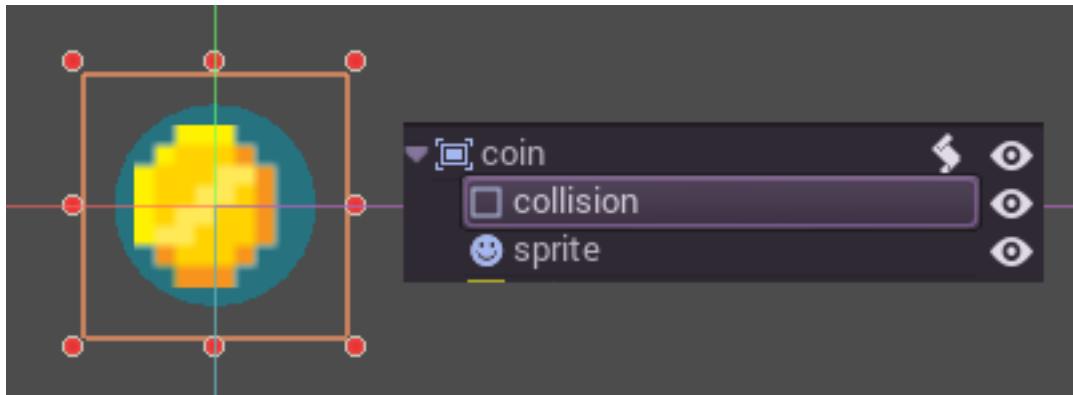
### CollisionShape2D

This node is a helper node. It must be created as a direct children of a **CollisionObject2D** derived node ([Area2D](#)).

By itself it does nothing, but when created as a child of the above mentioned nodes, it adds collision shapes to them. Any amount of **CollisionShape2D** children can be created, meaning the parent object will simply have more collision shapes. When added/deleted/moved/edited, it updates the list of shapes in the parent node.

At run time, though, this node does not exist (can't be accessed with `get_node()`), since it's only meant to be an editor helper. 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** with child **CollisionObject2D** and coin sprite:



## Triggers

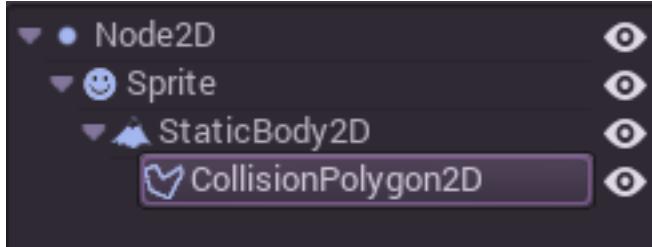
A CollisionShape2D or CollisionPolygon2D can be set as a trigger. When used in a RigidBody2D or KinematicBody2D, “trigger” shapes become non-collidable (objects can’t collide against it). They just move around with the object as ghosts. This makes them useful in two situations:

- Disabling collision in a specific shape.
- Get an Area2D to trigger a body\_enter / body\_exit signals with non collidable objects (useful in several situations).

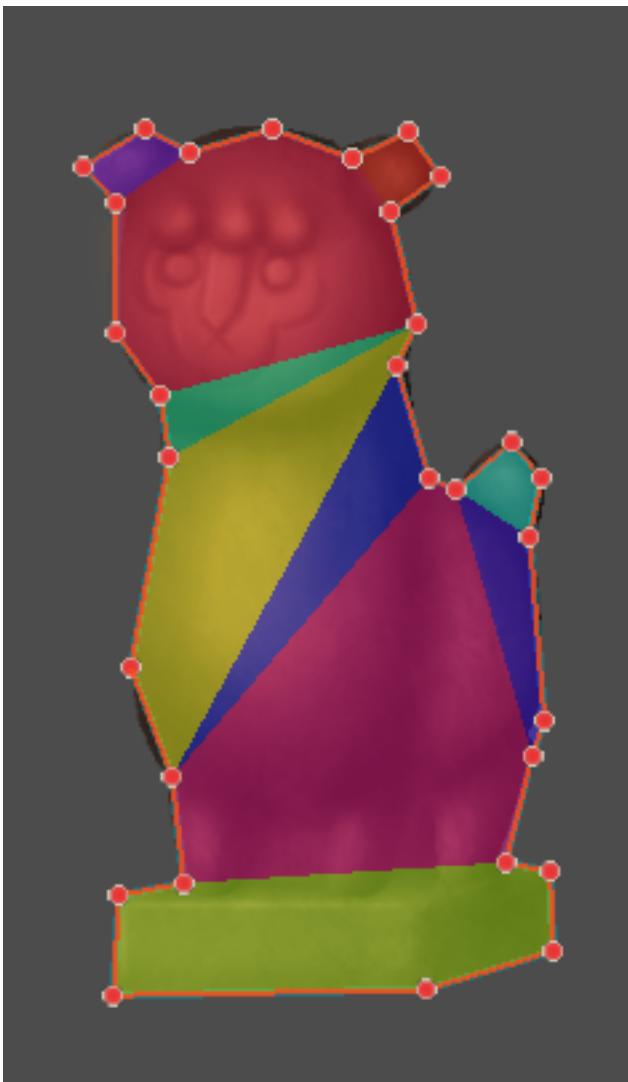
## CollisionPolygon2D

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.

Going back, here’s the scene with the StaticBody2D, the static body is the child of a sprite (meaning if the sprite moves, the collision does too). In turn, the CollisionPolygon is a child of staticbody, meaning it adds collision shapes to it.



In fact, what CollisionPolygon does is to decompose the polygon in convex shapes (shapes can only be convex, remember?) and adds them to the CollisionObject2D:



## KinematicBody2D

**Kinematic** bodies are special types of bodies that are meant to be user-controlled. They are not affected by the physics at all (to other types of bodies, such a character or a rigidbody, these are the same as a staticbody). 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). 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. A special [\[\[tutorial\\_kinematic\\_char\]\]](#).

## RigidBody2D

This type of body simulates newtonian physics. It has mass, friction, bounce, and the 0,0 coordinates simulates 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.

As a plus, as this is very common in games, it is possible to change a RigidBody2D node to behave like a Character (no rotation), StaticBody or KinematicBody according to different situations (example, an enemy frozen by an ice beam becomes a StaticBody)

The best way to interact with a RigidBody2D is during the force integration callback. In this very moment, the physics engine synchronizes state with the scene and allows full modification of the internal parameters (otherwise, as it may be running in a thread, changes will not take place until next frame). To do this, the following function must be overridden:

```
func _integrate_forces(state):
    [use state to change the object]
```

The ‘state’ parameter is of type [Physics2DDirectBodyState](#). Please do not use this object (state) outside the callback as it will result in an error.

## 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 (ie. 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:idx)
body_exit_shape(body_id:int,body:PhysicsBody2D,body_shape_index:int,area_shape_index:idx)
area_enter_shape(area_id:int,area:Area2D,area_shape_index:int,self_shape_index:idx)
area_exit_shape(area_id:int,area:Area2D,area_shape_index:int,self_shape_index:idx)

Areas also by default receive mouse/touchscreen input, providing a lower-level way than controls to ...

```

## Physics Global Variables

A few global variables can be tweaked in the project settings for adjusting how 2D physics works:

physics_2d		
INT	bp_hash_table_size	4096
INT	cell_size	128
INT	default_gravity	98
→	default_gravity_vector	0,1
REAL	default_density	0.1

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:

```

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!

## Contact Reporting

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:

### In case of overlap, who receives collision information?

<b>Type</b>	RigidBody	<i>CharacterBody</i>	KinematicBody	<i>StaticBody</i>	Area	
<b>RigidBody</b>	Both	Both	Both	Rigidbody	Area	
<b>CharacterBody</b>	Both	Both	Both	CharacterBody	Area	
<b>KinematicBody</b>	Both	Both	None	None	Area	
<b>StaticBody</b>	RigidBody	CharacterBody	None	None	None	
<b>Area</b>	Area	Area	Area	None	Both	

### 3.3.2 Kinematic Character (2D)

#### Introduction

Yes, the name sounds strange. “Kinematic Character” WTF 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.

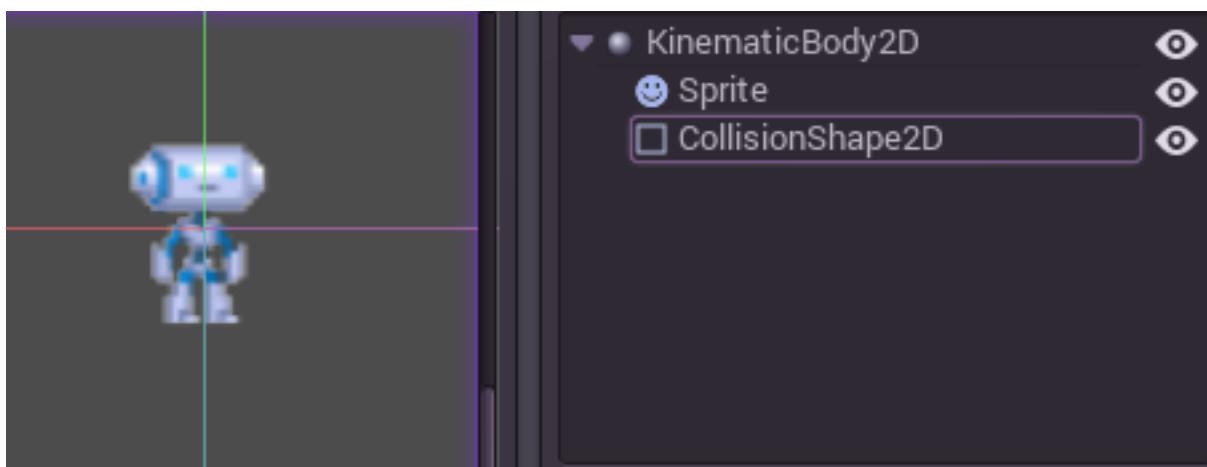
```
extends KinematicBody2D

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) attachment:kbscene.zip. We'll be creating a new scene for the character. Use the robot sprite and create a scene like this:



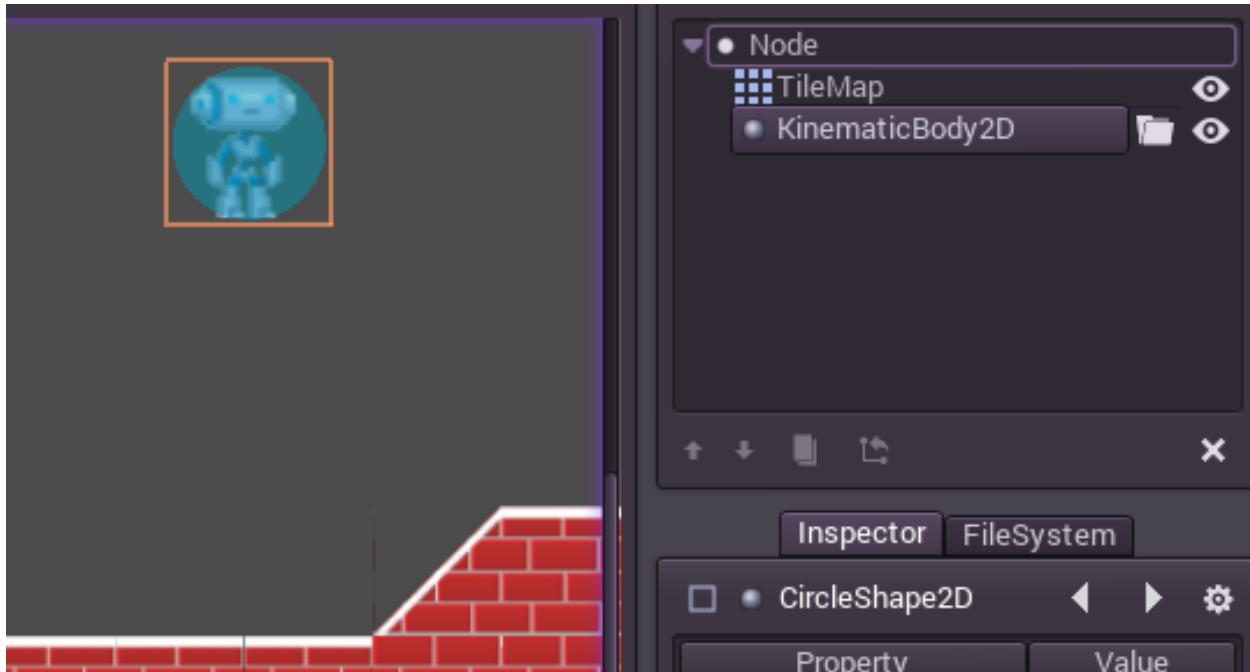
Let's add a circular collision shape to the collision body, create a new CircleShape2D in the shape property of CollisionShape2D. Set the radius to 30:



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 `move(motion_vector:Vector2)`. 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:

```
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:

```
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:

```
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)

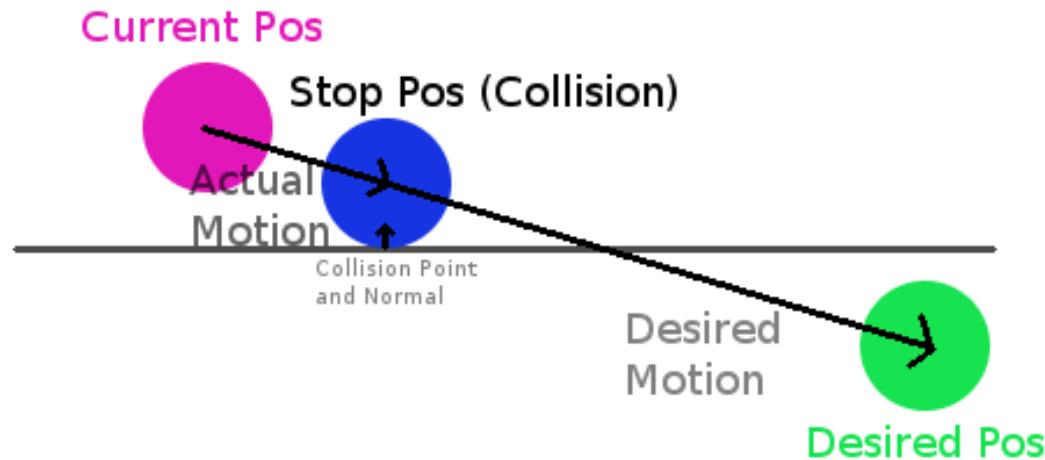
```

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:



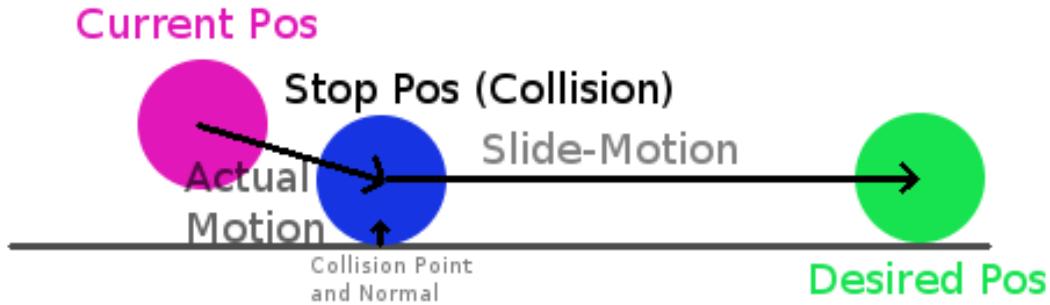
Basically, the desired motion vector will never complete because it hits the floor and the wall too early in the motion trajectory and that makes it stop there. Remember that even though the character is on the floor, the gravity is always turning the motion vector downwards.

### 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:

```
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/okamstudio/godot/tree/master/demos/2d/kinematic\\_char](https://github.com/okamstudio/godot/tree/master/demos/2d/kinematic_char).

### 3.3.3 Physics Ray Casting and Queries (2D and 3D)

#### Introduction

One of the most common tasks in game development is casting a ray (or custom shaped object) and see 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.

#### Acessing 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(delta)` 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:

```
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:

```
func _fixed_process(delta):
    var space_state = get_world_2d().get_direct_space_state()
```

For 3D:

```
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:

```
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 ray didn't hit anything, the dictionary will be empty. If it did hit something it will contain collision information:

```
if (not result.empty()):
    print("Hit at point: ",result.position)
```

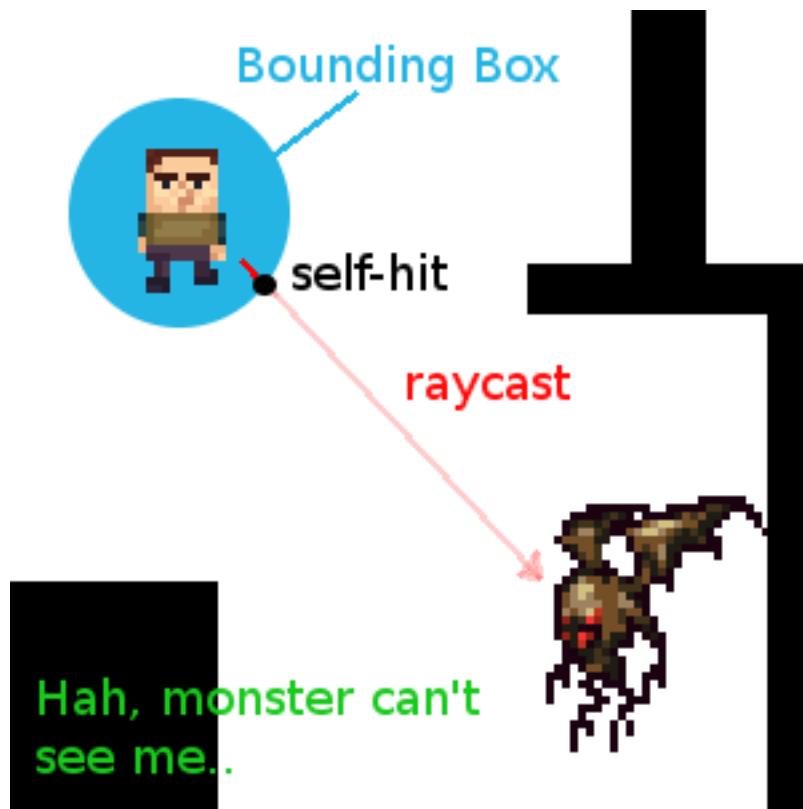
The collision result dictionary, when something hit, has this format:

```
{
    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:

```
extends KinematicBody2D

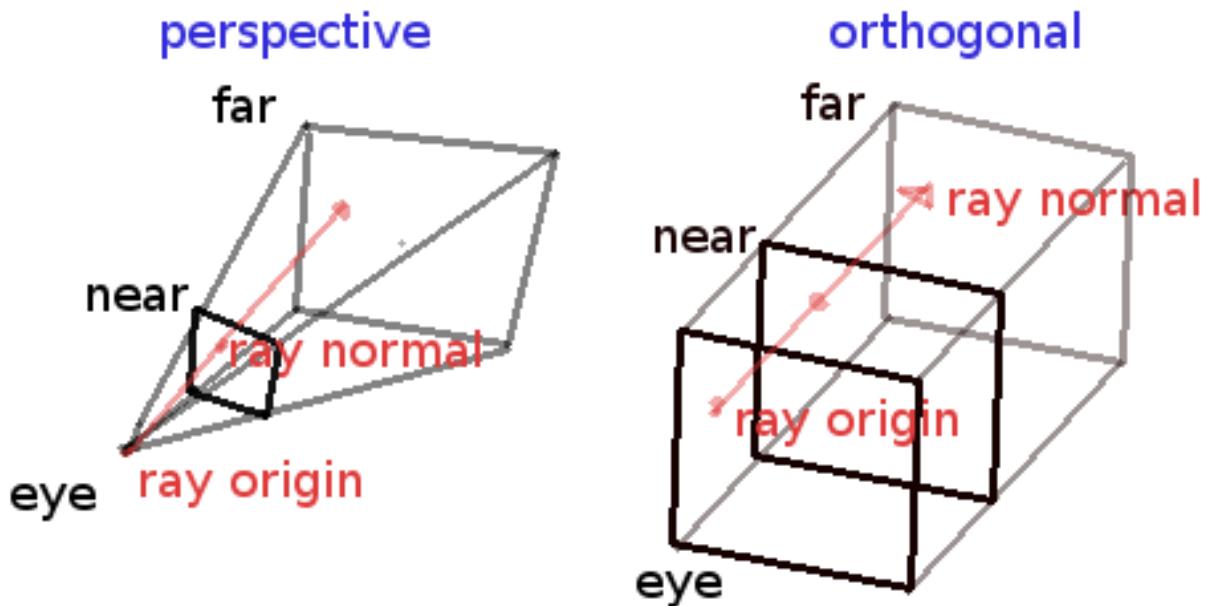
func _fixed_process(delta):
    var space_state = get_world().get_direct_space_state()
    var result = space_state.intersect_ray( get_global_pos(), enemy_pos, [ self ] )
```

The extra argument is a list of exceptions, can be objects (need Godot 1.1beta2+ for this) 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:

```
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()`.



---

## 3D tutorials

---

### 4.1 Graphics

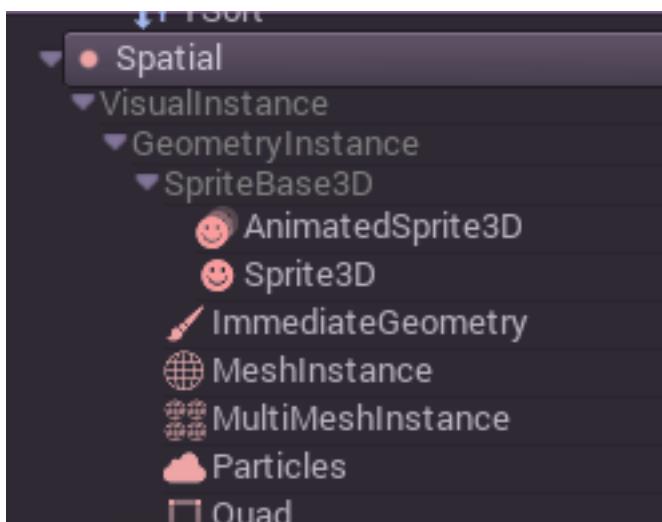
#### 4.1.1 Introduction

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.

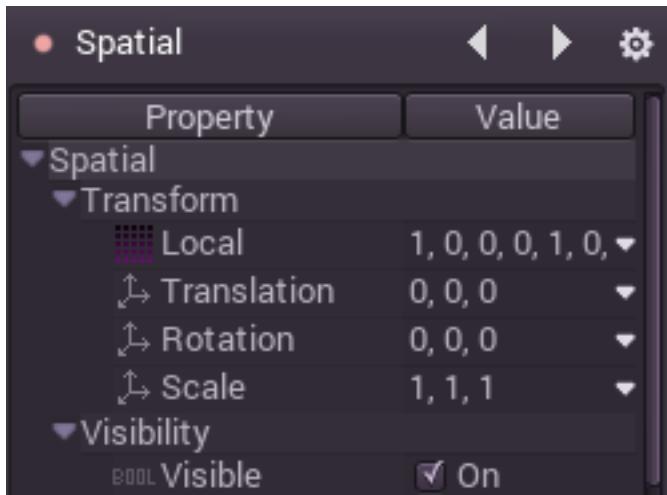
In 3D, math is a little more complex than in 2D, so also checking the [[Vector Math]] in the wiki (which were specially created for game developers, not mathematicians or engineers) will help pave the way into efficiently developing 3D games.

#### 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 [[Import 3D]] 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 [[Import 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

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

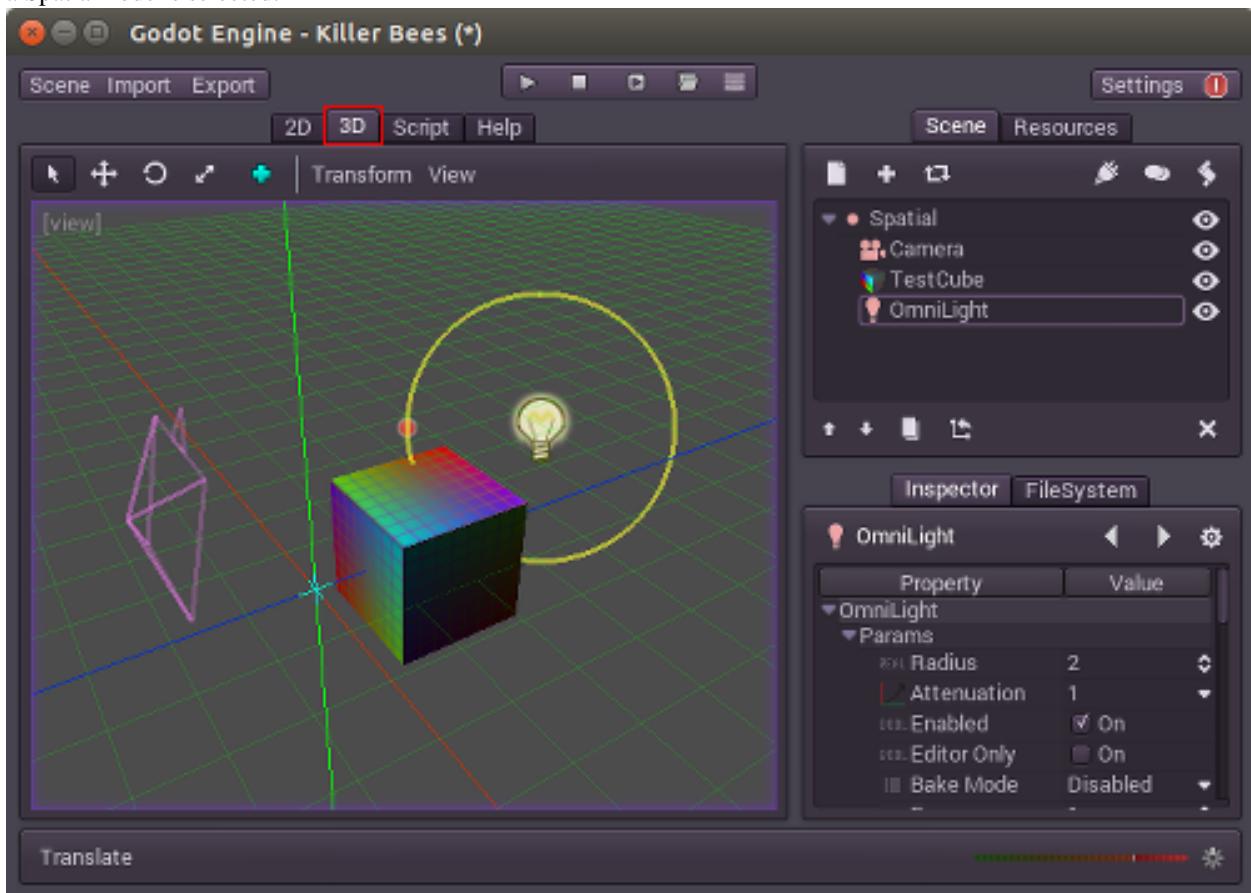
The disadvantage is, of course, that added complexity and reduced performance in comparison to plain 2D, as well as the lack of reference of working in pixels.

## Environment

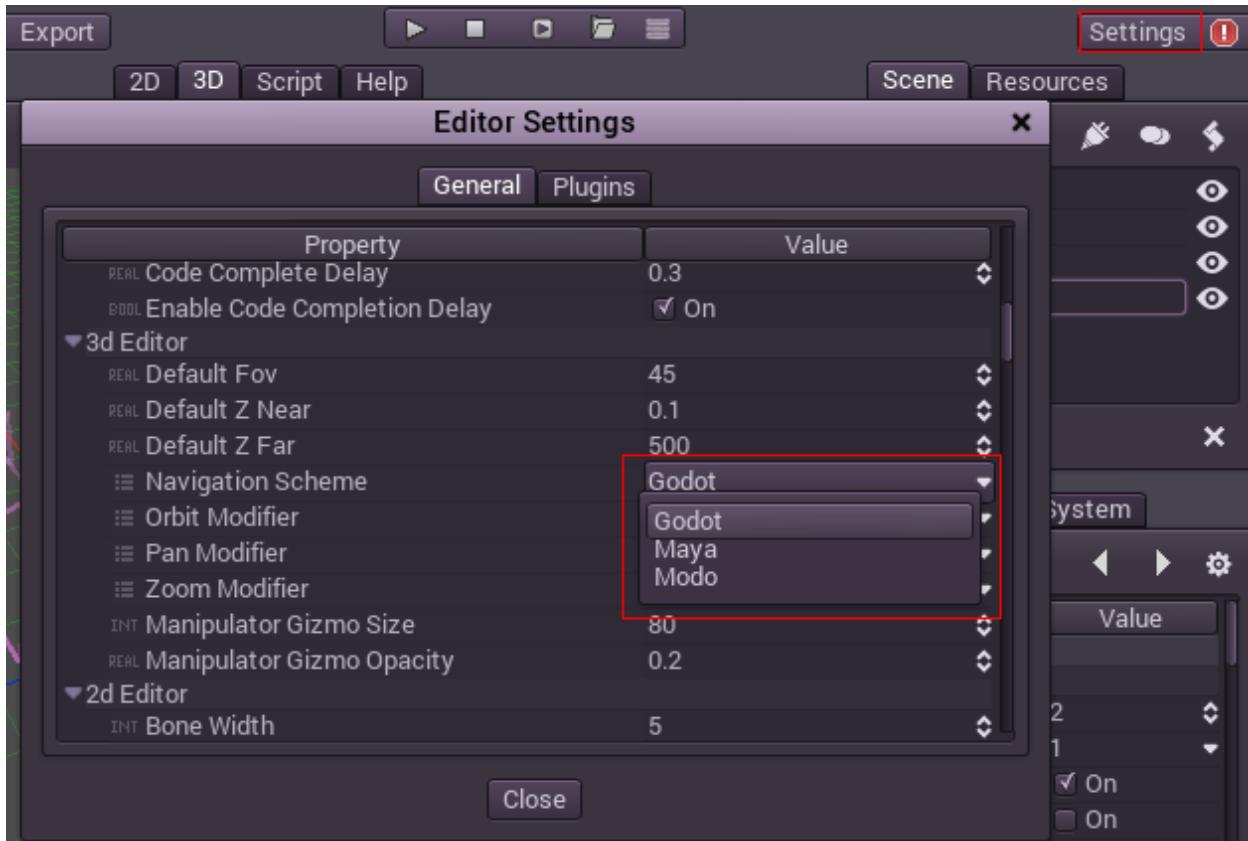
Besides editing a scene, it is often common to edit the environment. Godot provides a [WorldEnvironment](#) node that allows changing the background color, mode (as in, put a skybox), and applying several types of built-in post-processing effects. Environments can also be overridden in the Camera.

## 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.



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).

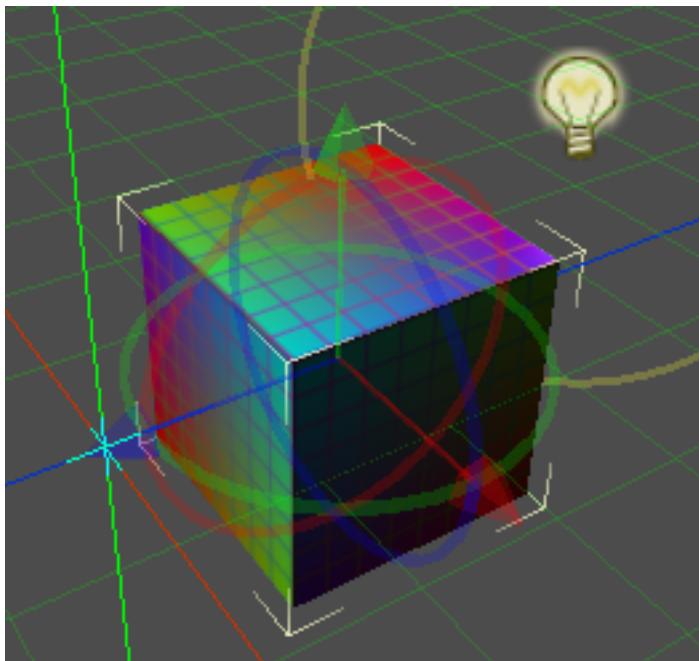
When working with 3D assets, it's always best to work in the correct scale (set your DCC to metric). Godot allows scaling post-import and, while this works in most cases, in rare situations it may introduce floating point precision issues (and thus, glitches or artifacts) in delicate areas such as rendering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

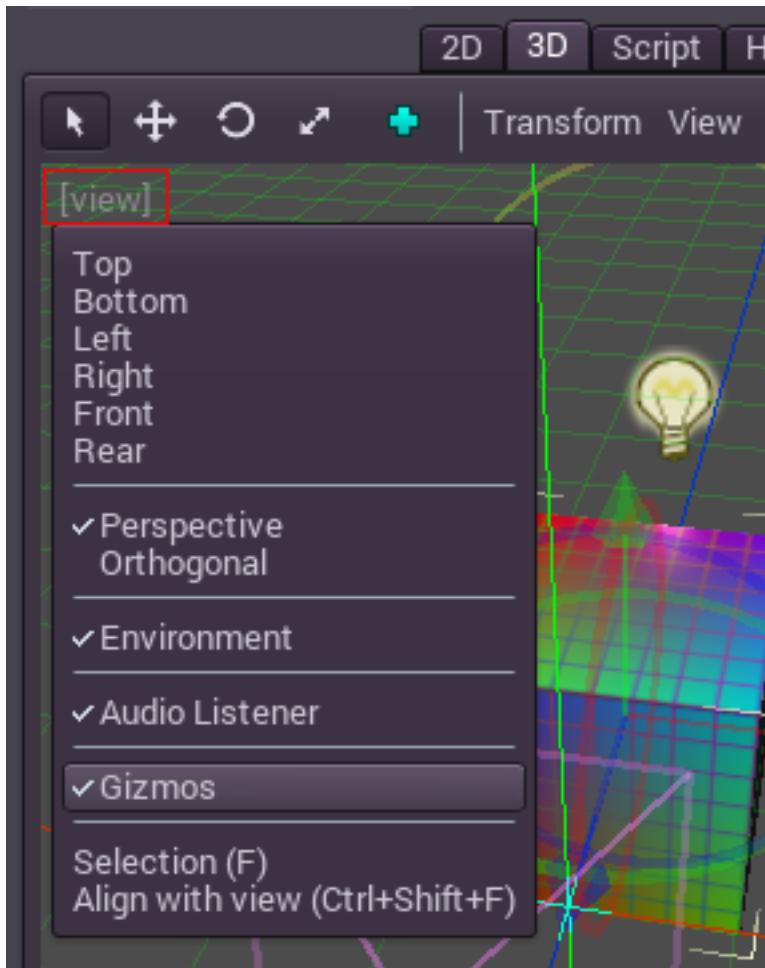


Some useful keybindings:

- 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!

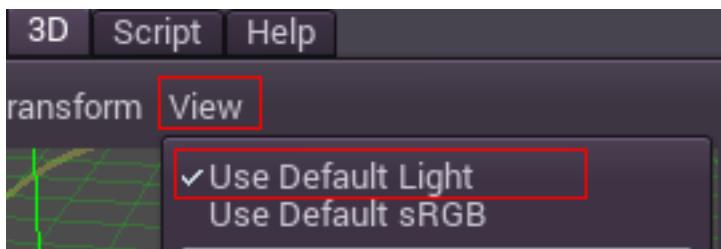


### Default Lighting

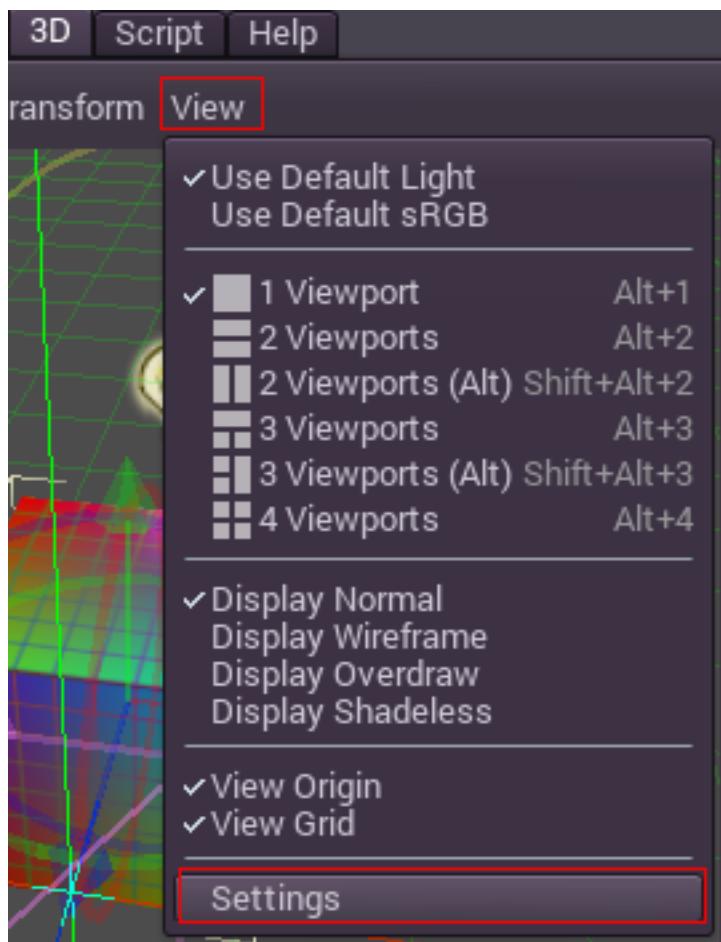
The 3D View has some default options on lighting:

- 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:

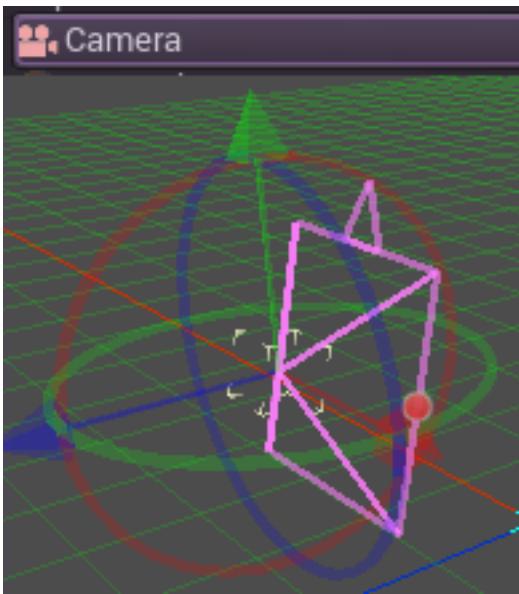


which opens this window, allowing to customize ambient light color and default light direction:

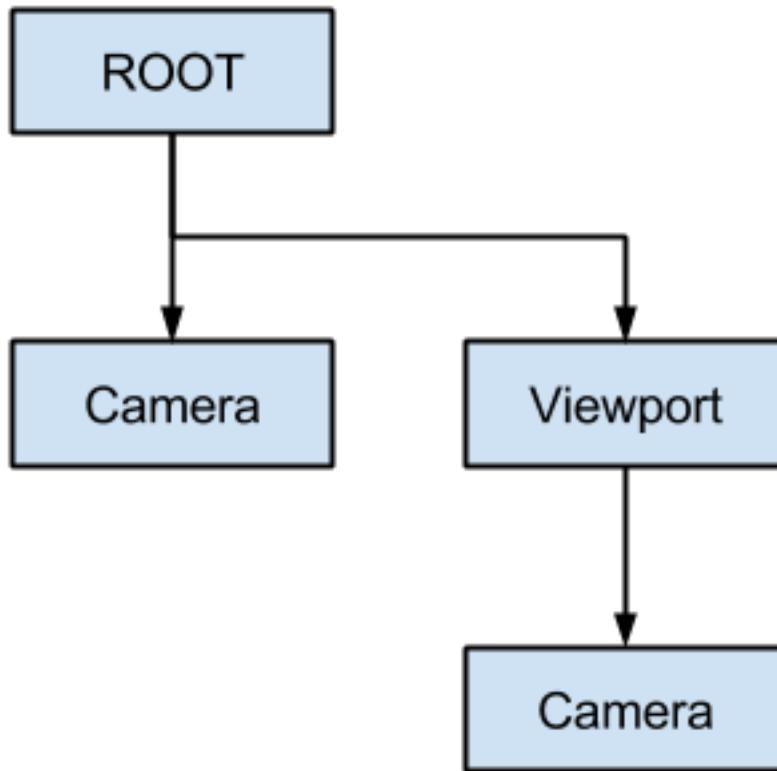


## Cameras

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-viewports (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 and types 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.

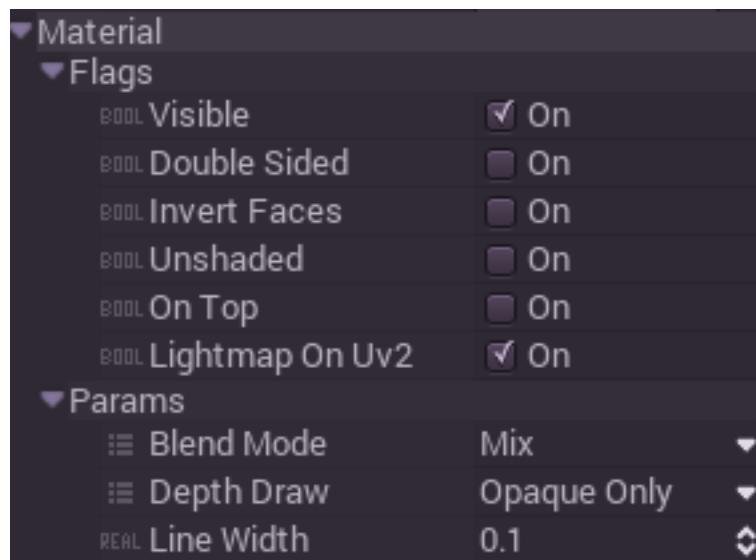
It is possible to use [[Light Baking]], to avoid using large amount of real-time lights and improve performance.

### 4.1.2 Materials

#### Introduction

Materials can be applied to most visible 3D objects, they basically are a description to how light reacts to that object. There are many types of materials, but the main ones are the [FixedMaterial](#) and [ShaderMaterial](#). Tutorials for each of them exist [[Fixed Material]] and [[Shader Material]].

This tutorial is about the basic properties shared between them.



#### Flags

Materials, no matter which type they are, have a set of flags associated. Each has a different use and will be explained as follows.

##### Visible

Toggles whether the material is visible. If unchecked, the object will not be shown.

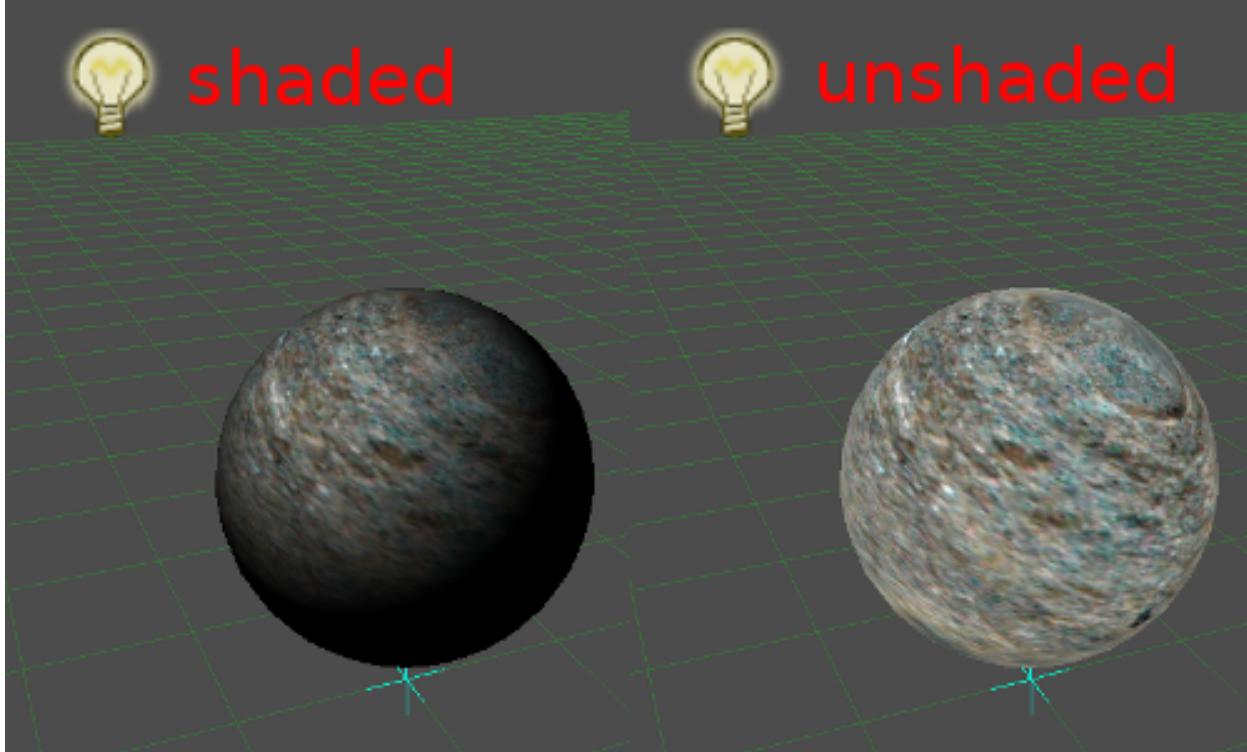
## Double Sided & Invert Faces

Godot by default only shows geometry faces (triangles) when facing the camera. To do this it needs them to be in view in clockwise order. This saves a lot of GPU power by ensuring that not visible triangles are not drawn.

Some flat objects might need to be drawn all the times though, for this the “double sided” flag will make sure that no matter the facing, the triangle will always be drawn. It is also possible to invert this check and draw counter-clockwise looking faces too, 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:



## 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 only useful for HUD effects or gizmos.

## Lightmap on UV2

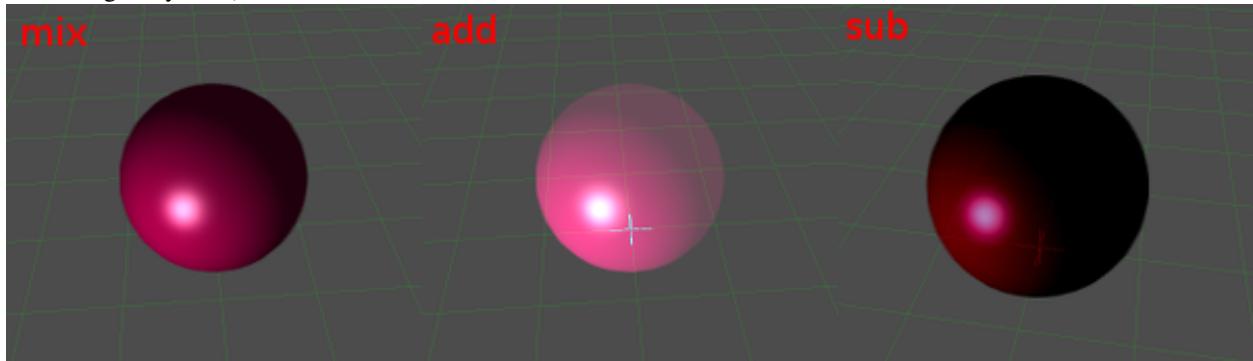
When using lightmapping (see the [[Light Baking]] tutorial), this option determines that the lightmap should be accessed on the UV2 array instead of regular UV.

## 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:



## 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 use 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, specially 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.



### 4.1.3 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:

Property	Value
<b>FixedMaterial</b>	
<b>Fixed Flags</b>	
BOOL Use Alpha	<input checked="" type="checkbox"/> On
BOOL Use Color Array	<input checked="" type="checkbox"/> On
BOOL Use Point Size	<input checked="" type="checkbox"/> On
BOOL Discard Alpha	<input checked="" type="checkbox"/> On
<b>Params</b>	
Diffuse	
Specular	
Emission	
REAL Specular Exp	40
Detail Blend	Mix
REAL Detail Mix	1
REAL Normal Depth	1
REAL Shade Param	0.5
REAL Glow	0
REAL Point Size	1
Blend Mode	Mix
REAL Line Width	0.1
UV Xform	1, 0, 0, 0, 1, 0,
<b>Textures</b>	
Diffuse	<null>
Diffuse Tc	UV
Detail	<null>
Detail Tc	UV
Specular	<null>
Specular Tc	UV
Emission	<null>
Emission Tc	UV
Specular Exp	<null>
Specular Exp T	UV
Glow	<null>
Glow Tc	UV
Normal	<null>
Normal Tc	UV
Shade Param	<null>
Shade Param T	UV

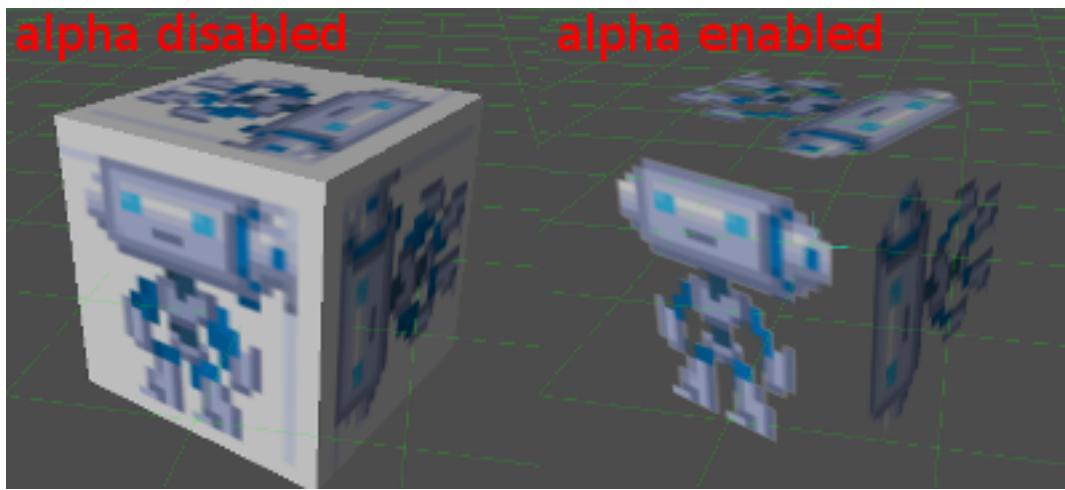
From this point, every option will be explained in detail:

### Fixed Flags

These are a set of flags that control general aspects of the material.

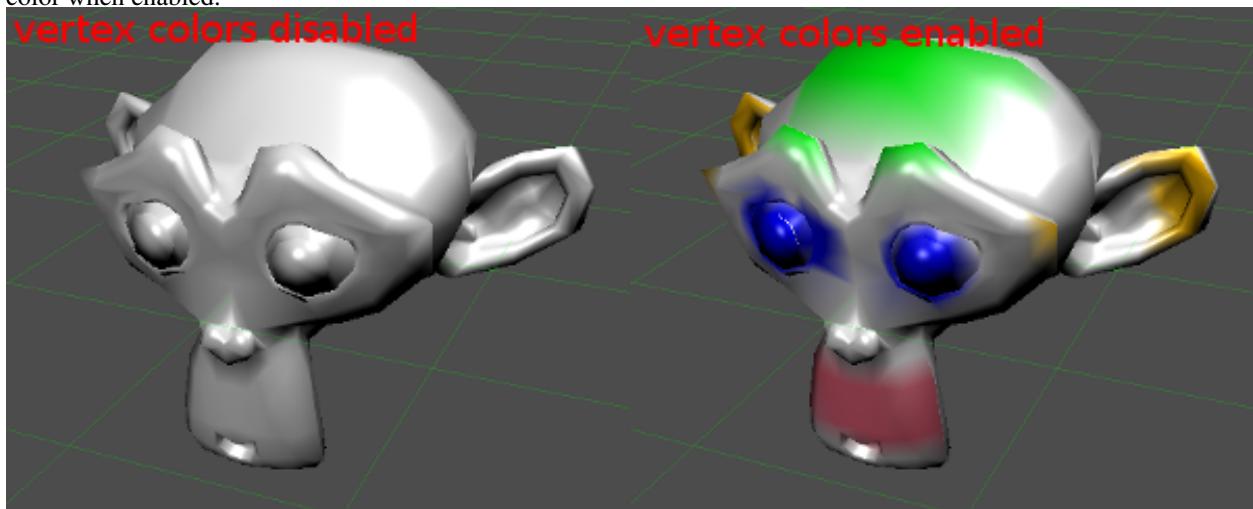
#### Use Alpha

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 will also not cast shadows (unless they contain opaque areas and the “opaque pre-pass” hint is turned on, see the [[Materials]] tutorial for more information).



#### Use Vertex Colors

Vertex color painting is a very common technique to add detail to 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.



## 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 invisible pixels are blended with what is behind them. In some combinations (of using alpha to render depth) it may be possible that invisible 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).

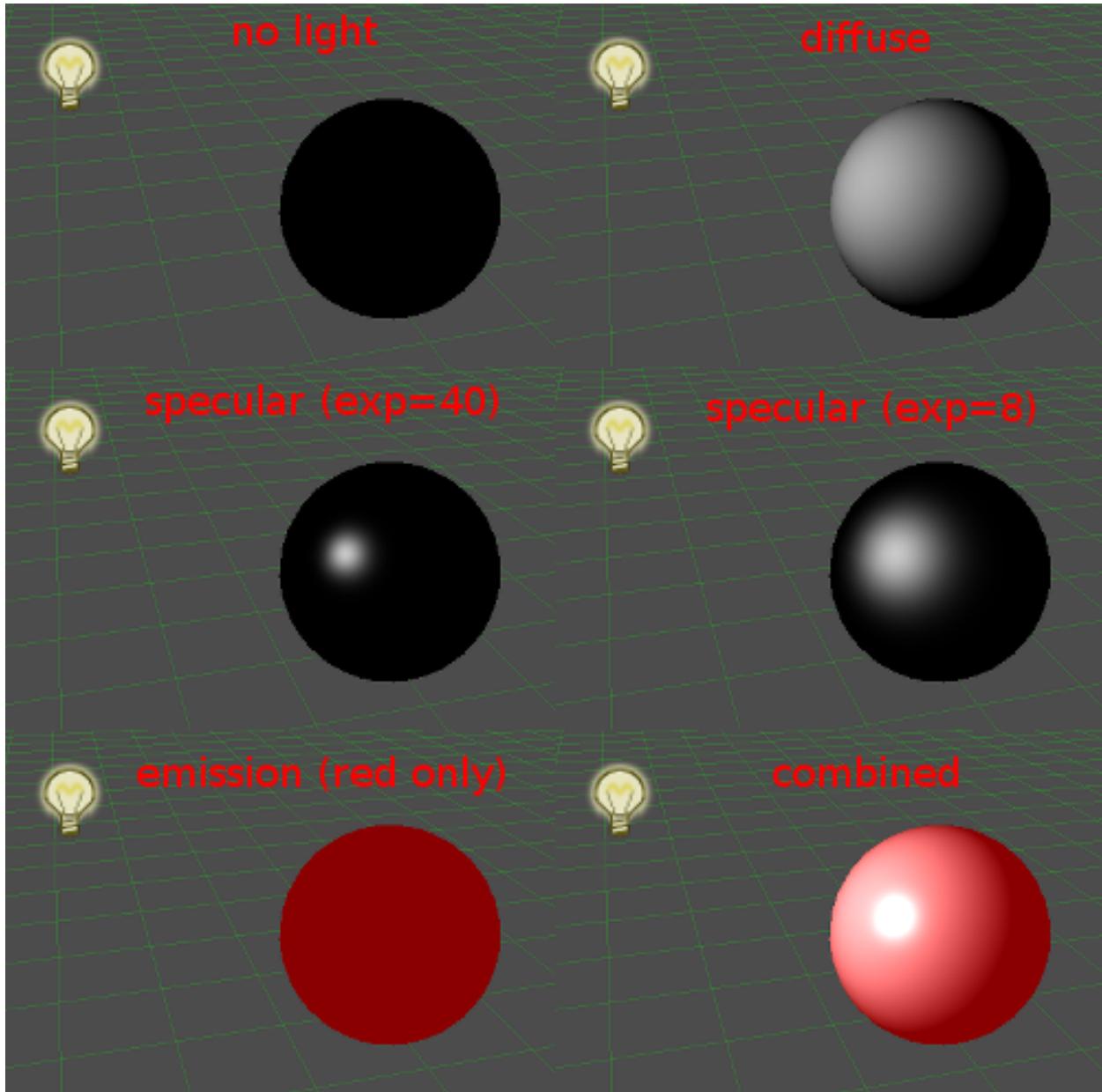
## Parameters

### 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, then gets diffused around. This color varies by the angle to the light and the distance (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.

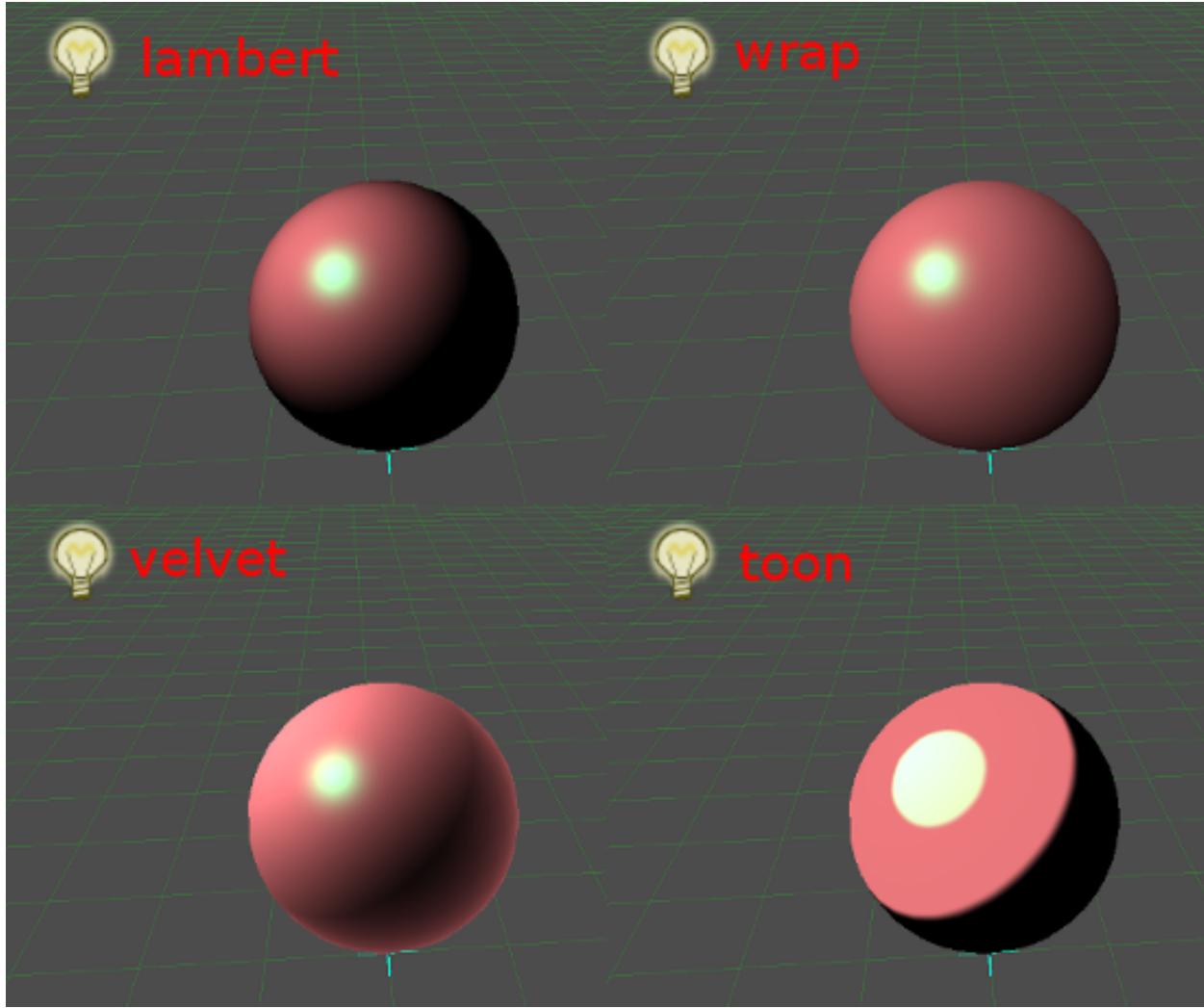
Below is an example of how they interact:



## 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 with the light emissor.
- **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.



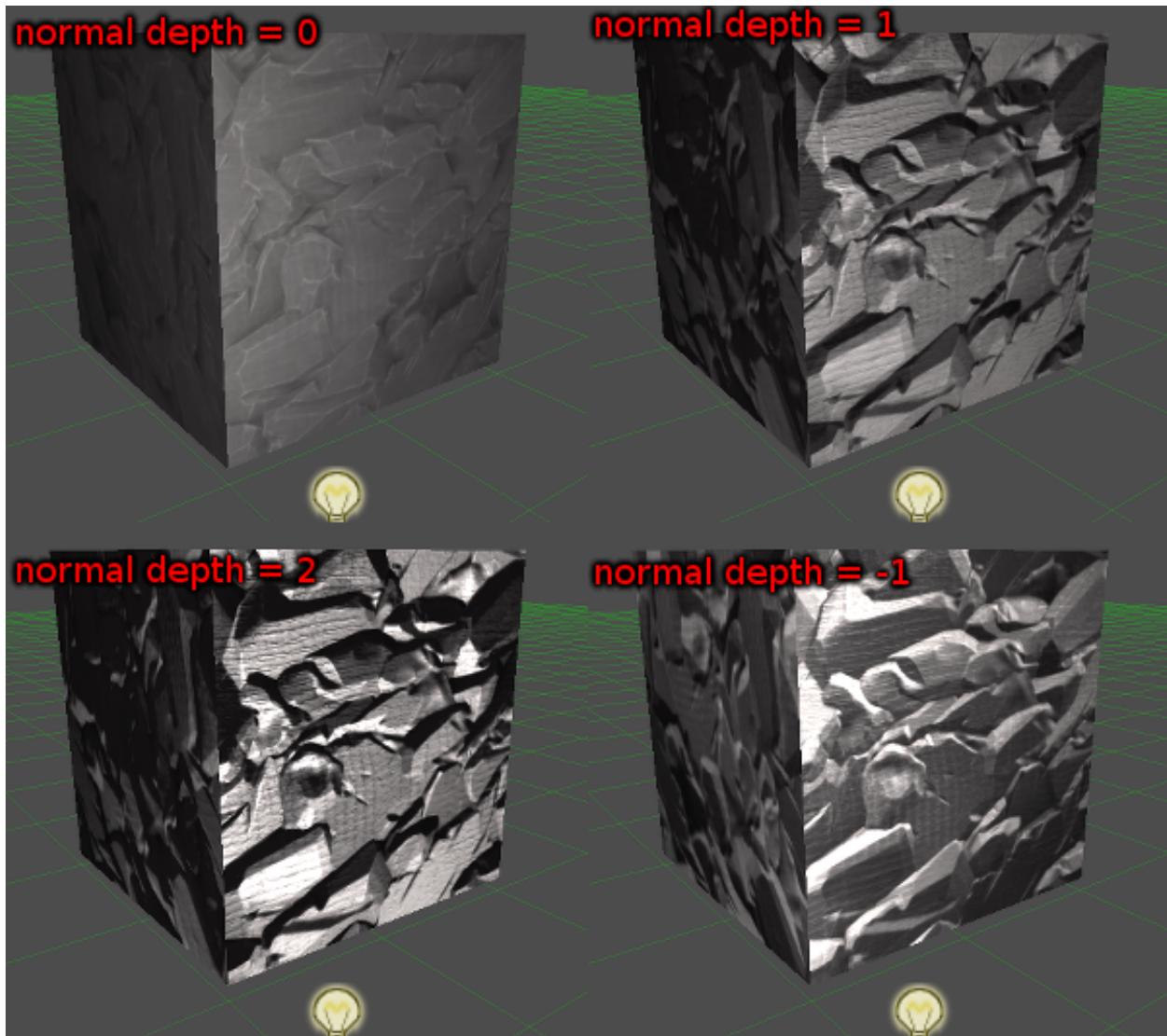
#### Detail & Detail Mix

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:



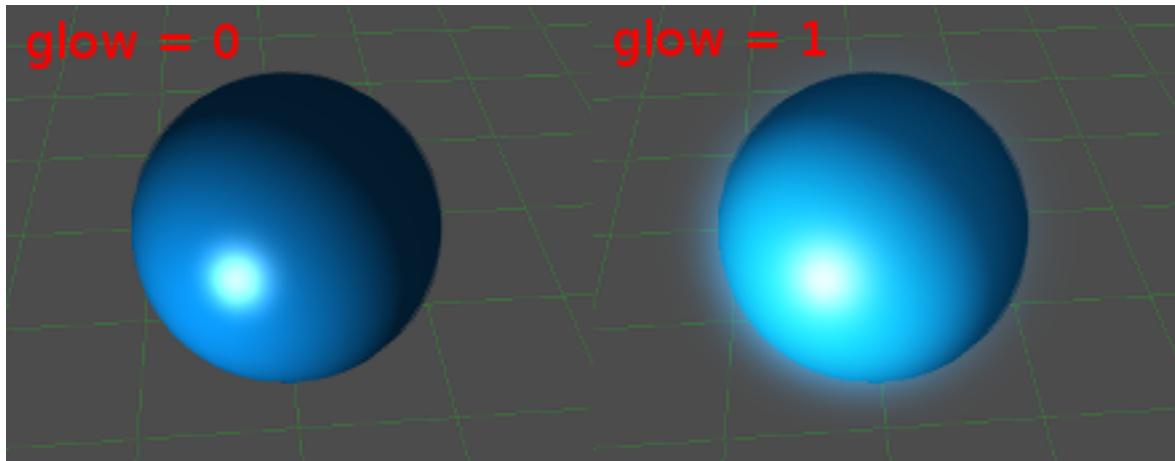
### Normal Depth

Normal depth controls the intensity of the normal-mapping as well as the direction. On 1 (the default) normalmapping applies normally, on -1 the map is inverted and on 0 is disabled. Intermediate or greater values are accepted. Here's how it's supposed to look:



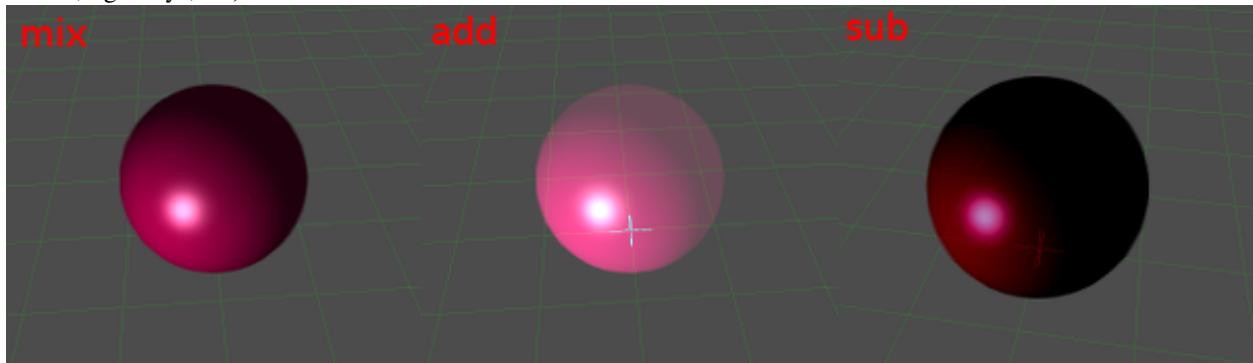
### Glow

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.

[[<https://creativecommons.org/licenses/by/3.0/legalcode>]] license.

## 4.1.4 Shader Materials

### Introduction

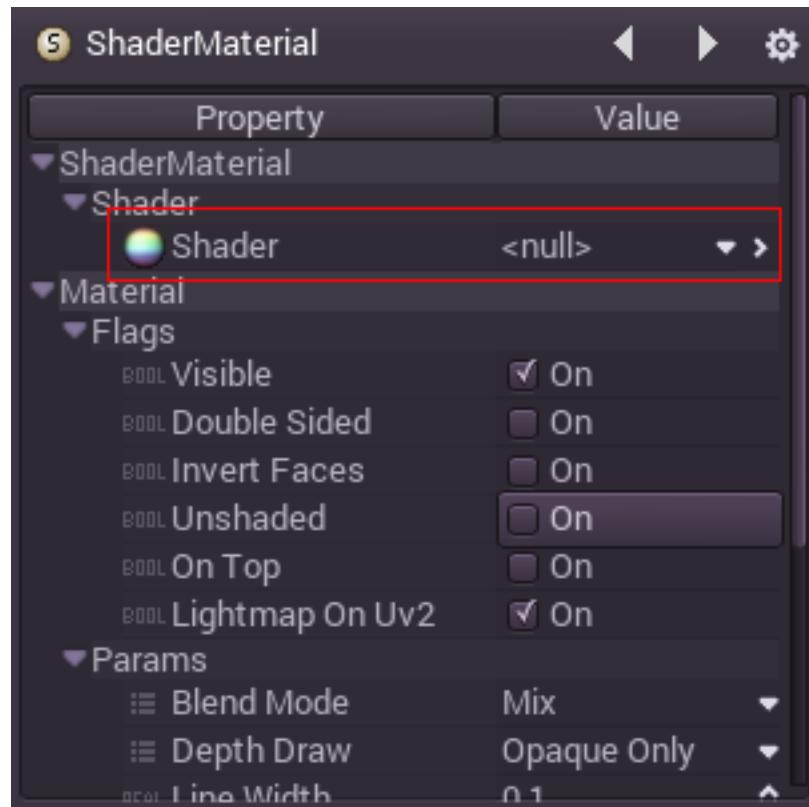
For the most common cases, [[Fixed Material]] are enough to create the desired textures or look and feel. Shader materials are a step beyond that adds a huge amount of flexibility. With them, it is possible to:

- 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 graph editor (NOTE: Currently disabled! work in progress!).

### Creating a ShaderMaterial

Create a new ShaderMaterial in some object of your choice. Go to the “Shader” property, then create a new “Shader”:



Edit the newly created shader, and the shader editor will open:

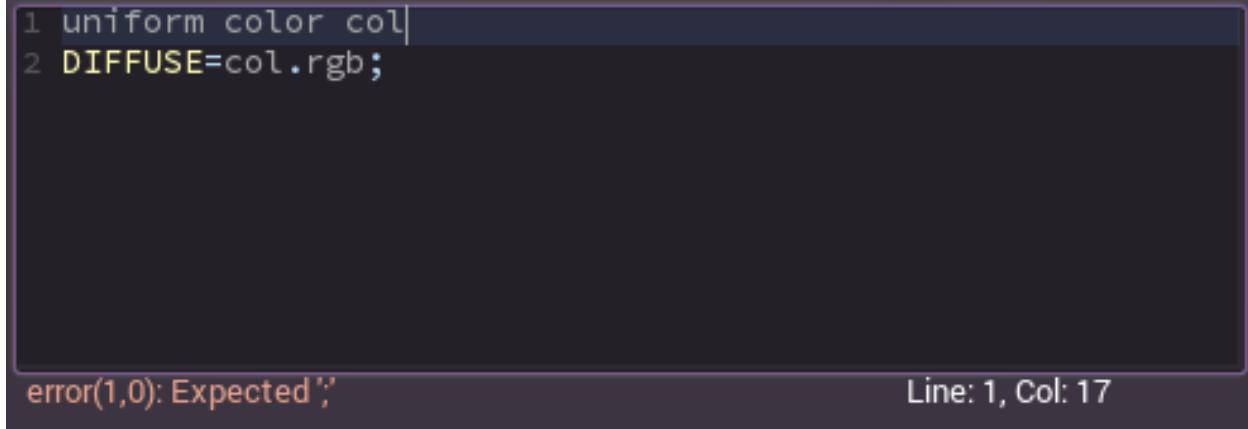


There are three code tabs open, the first is for the vertex shader, the second for the fragment and the third for the lighting. The shader language is documented in it's [[Shader]] so a small example will be presented next.

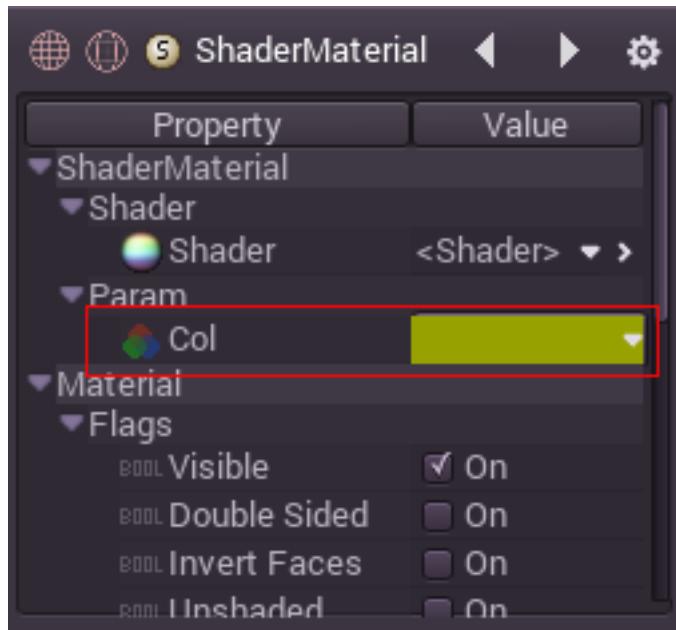
Create a very simple fragment shader that writes a color:

```
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.

## 4.1.5 Lighting

### Introduction

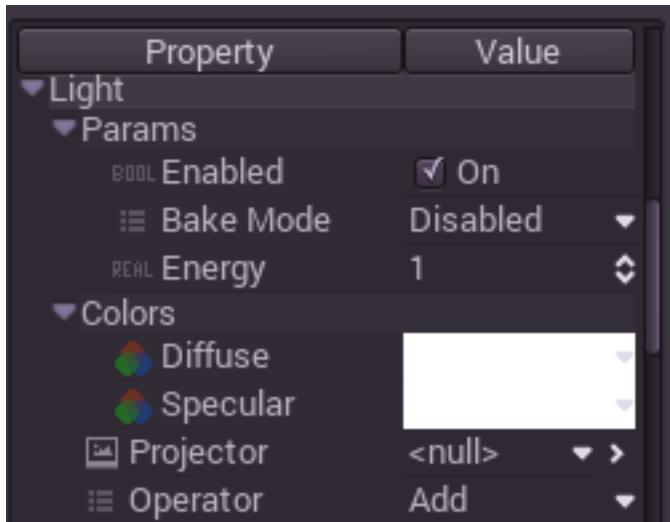
Lights emit light that mix with the materials and produces a visible result. Light can come from several types of sources in a scene:

- 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 [\[\[Light Baking\]\]](#)).

The emission color is a material property, as seen in the previous tutorials about materials (go read them if you didn't at this point!).

### Light Nodes

As mentioned before, there are three types of light nodes: Directional, Ambient 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:

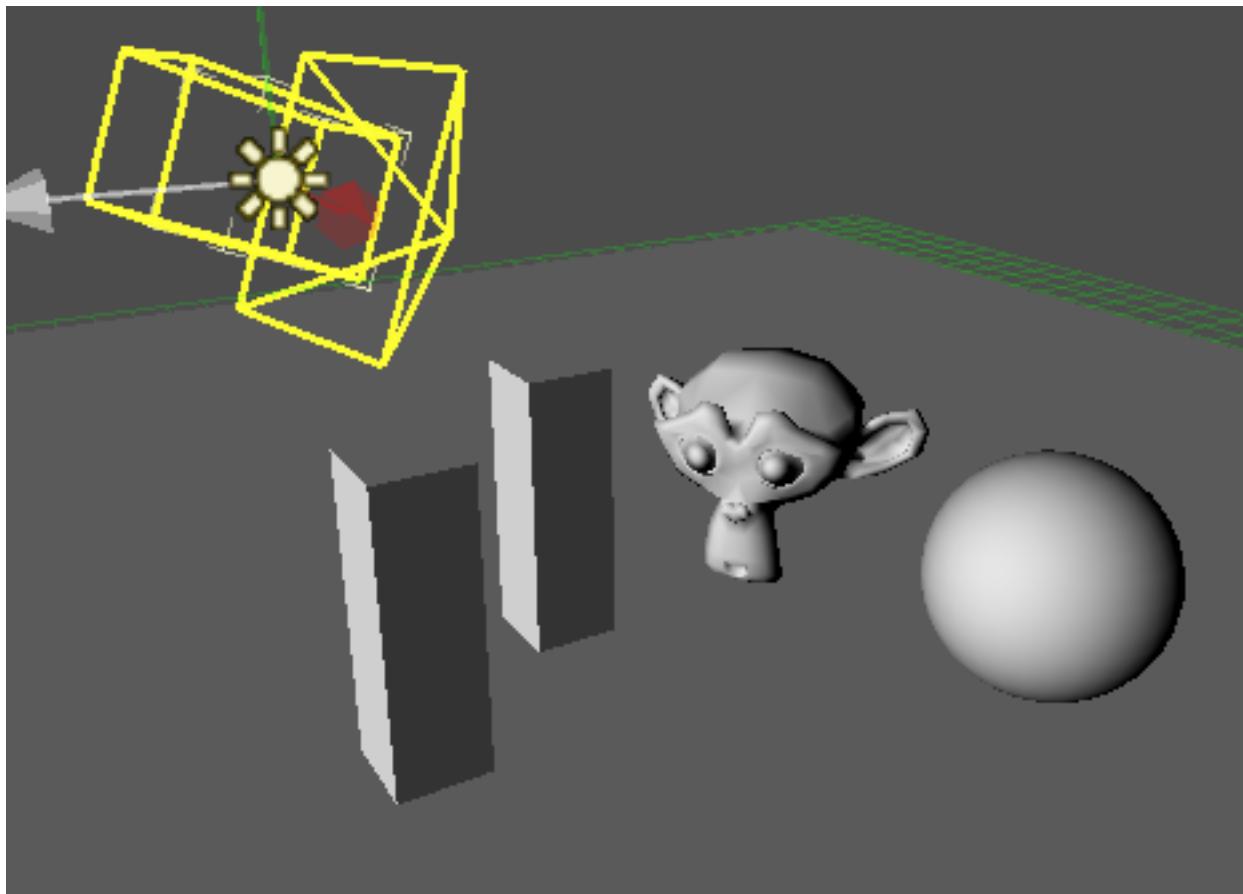


Each one has a specific function:

- **Enabled:** Lights can be disabled at any time.
- **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 [[HRD]] and for Spot and Omni lights, because it can create very bright spots near the emissor.
- **Diffuse and Specular:** These light values get multiplied by the material light and diffuse colors, so a white value does not mean that light will be white, but that the original 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

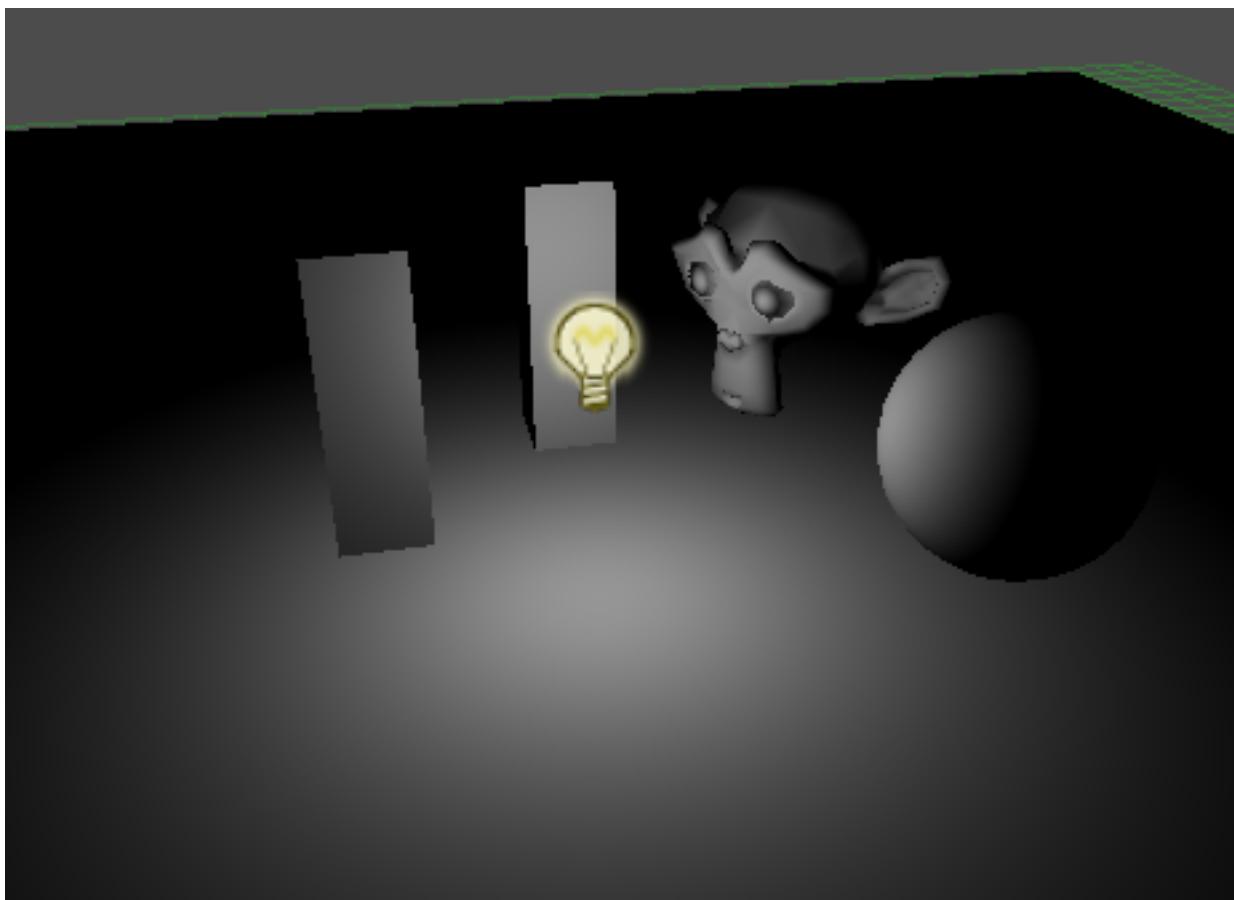
This is the most common type of light and represents the sun. It is also the cheapest light to compute and should be used whenever possible (although it's not the cheapest shadow-map to compute, but more on that later). Directional light nodes are represented by a big arrow, which represent the direction of the light, however the position of the node does not affect the lighting at all, and can be anywhere.



Basically what faces the light is lit, what doesn't is dark. Most lights have specific parameters but directional lights are pretty simple in nature so they don't.

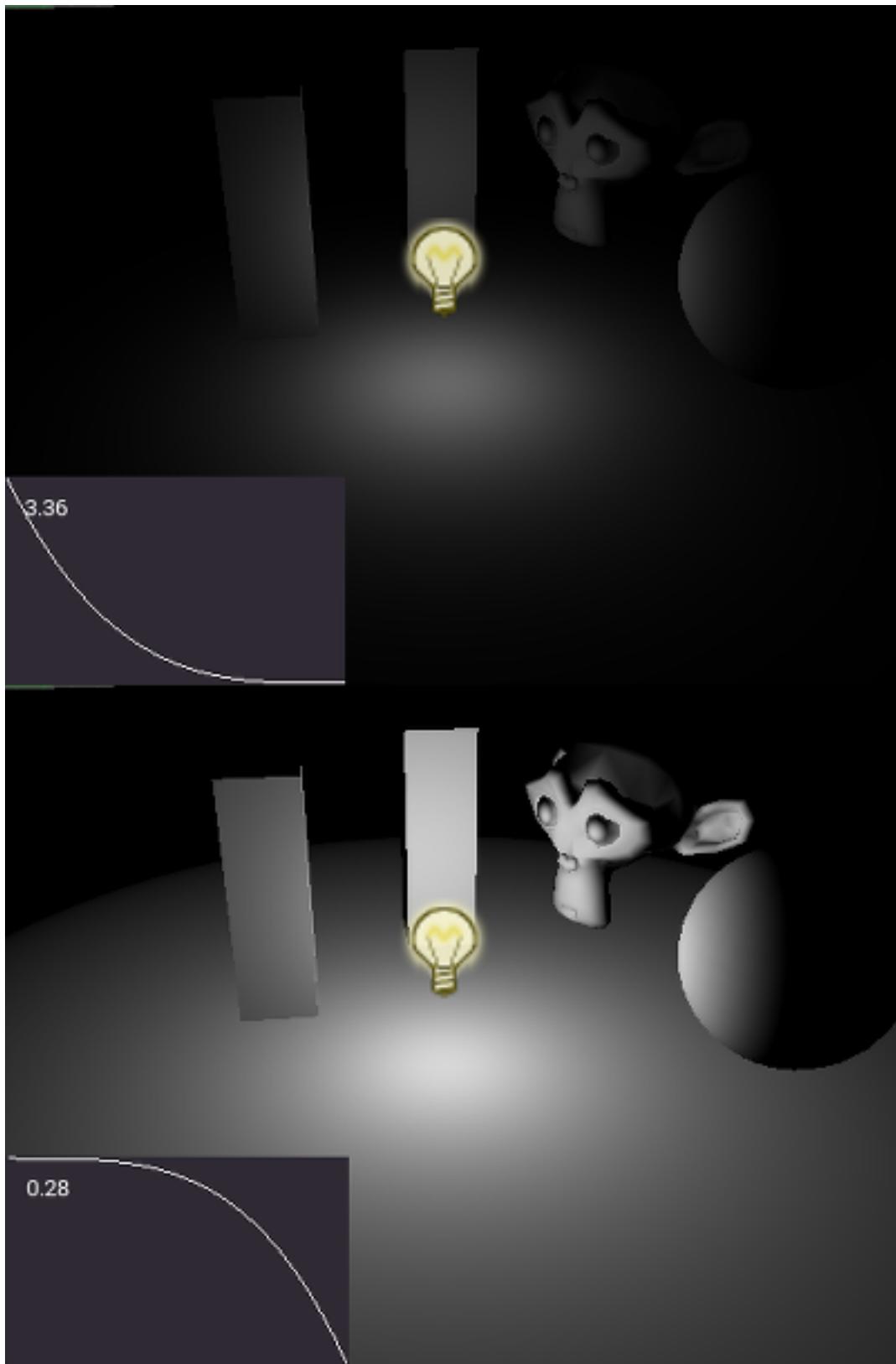
### Omni Light

Omni light is a point that throws light all around it up to a given radius (distance) that can be controlled by the user. The light attenuates with the distance and reaches 0 at the edge. It represents lamps or any other light source that comes from a point.



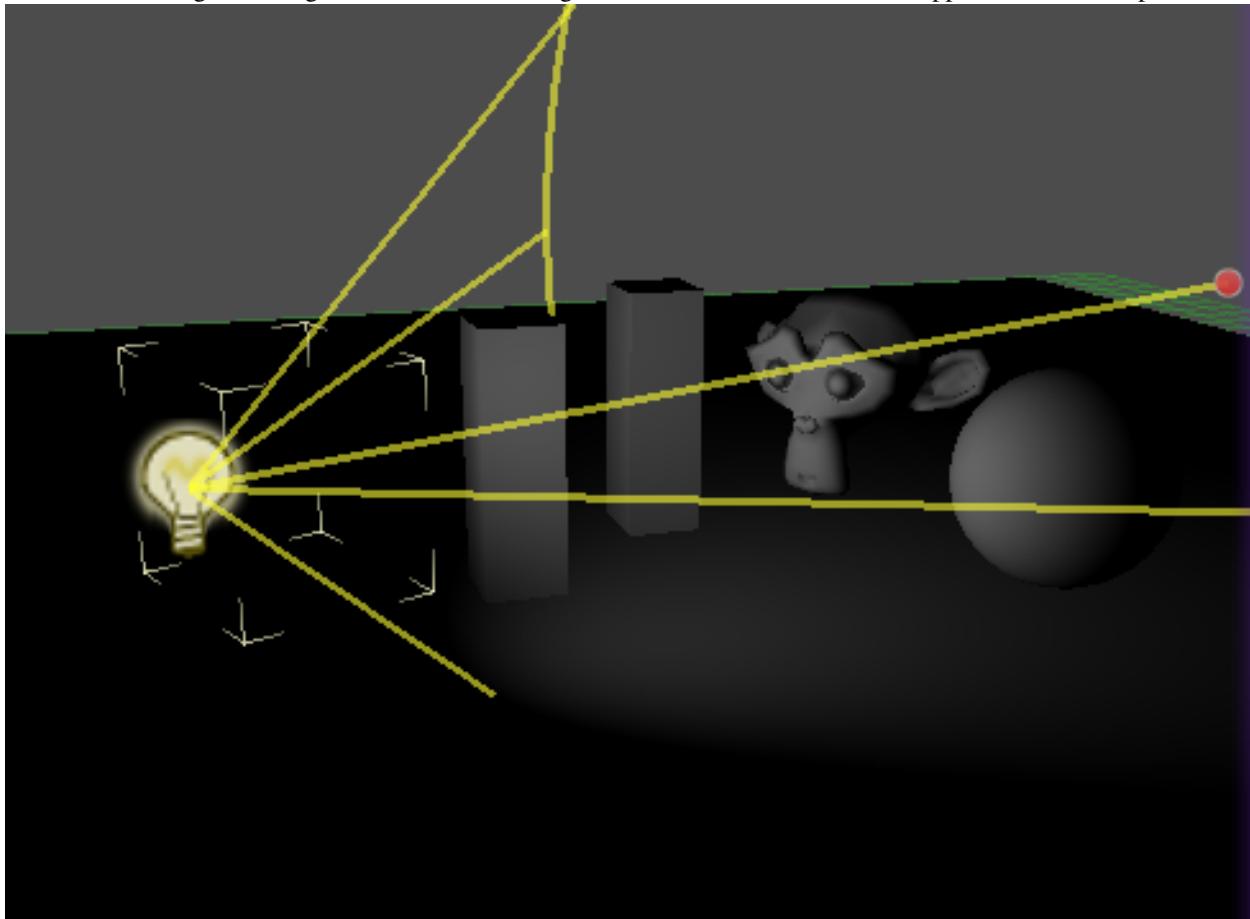
The attenuation curve for these kind of lights in nature is computed with an inverse-quadratic function that never reaches zero and has almost infinitely large values near the emissor.

This makes them considerably inconvenient to tweak for artists, so Godot simulates them with an artist-controlled exponential curve instead.



## Spot Light

Spot lights are similar to Omni lights, except they only operate between a given angle (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 [Baked Light](#) tutorial for more information).

## 4.1.6 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".

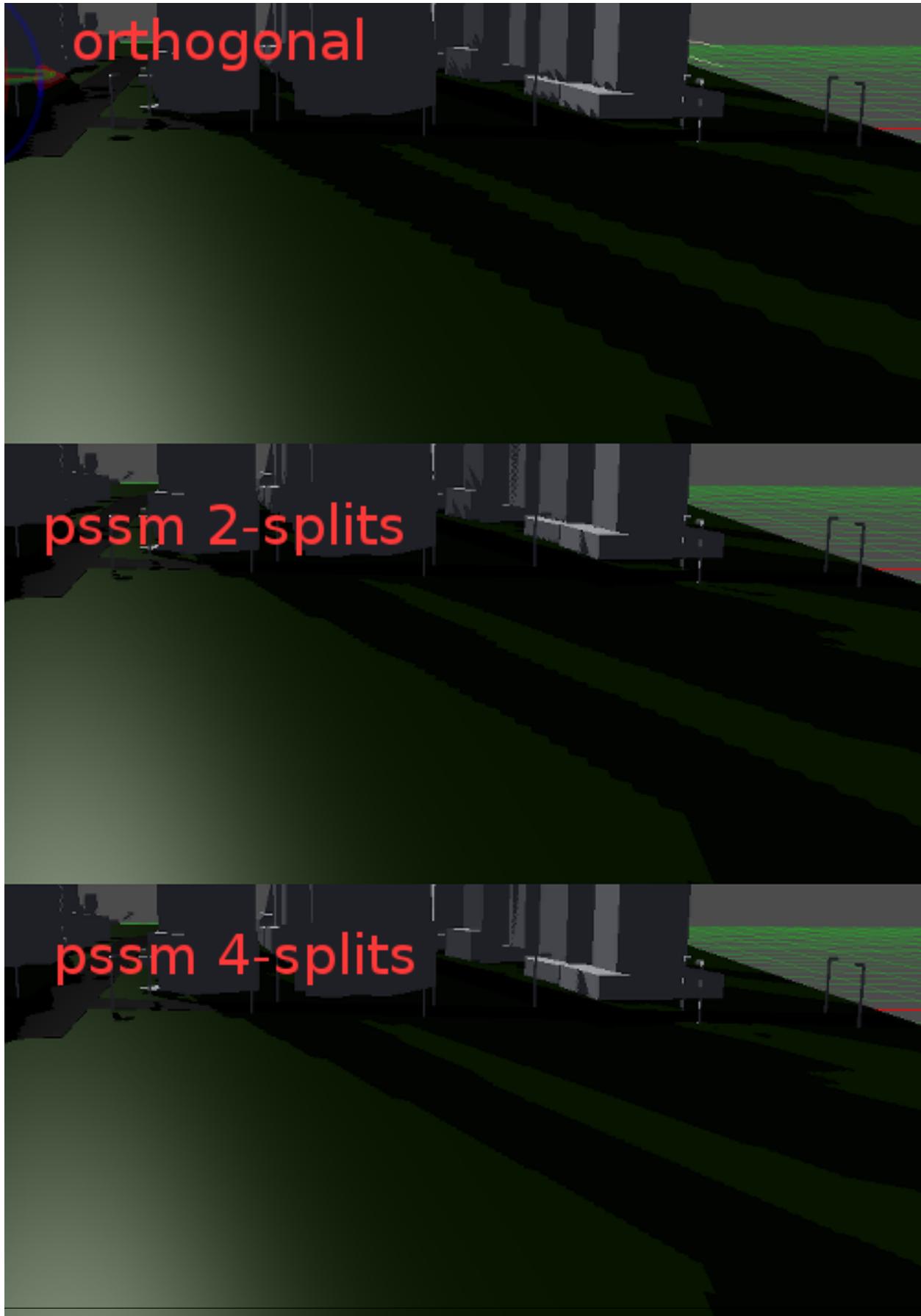
All light types in Godot can use shadow mapping, and all support several different techniques that trade quality by performance. Shadow mapping uses a texture storing the "depth view" of the light and checks against it in real-time for each pixel it renders.

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):

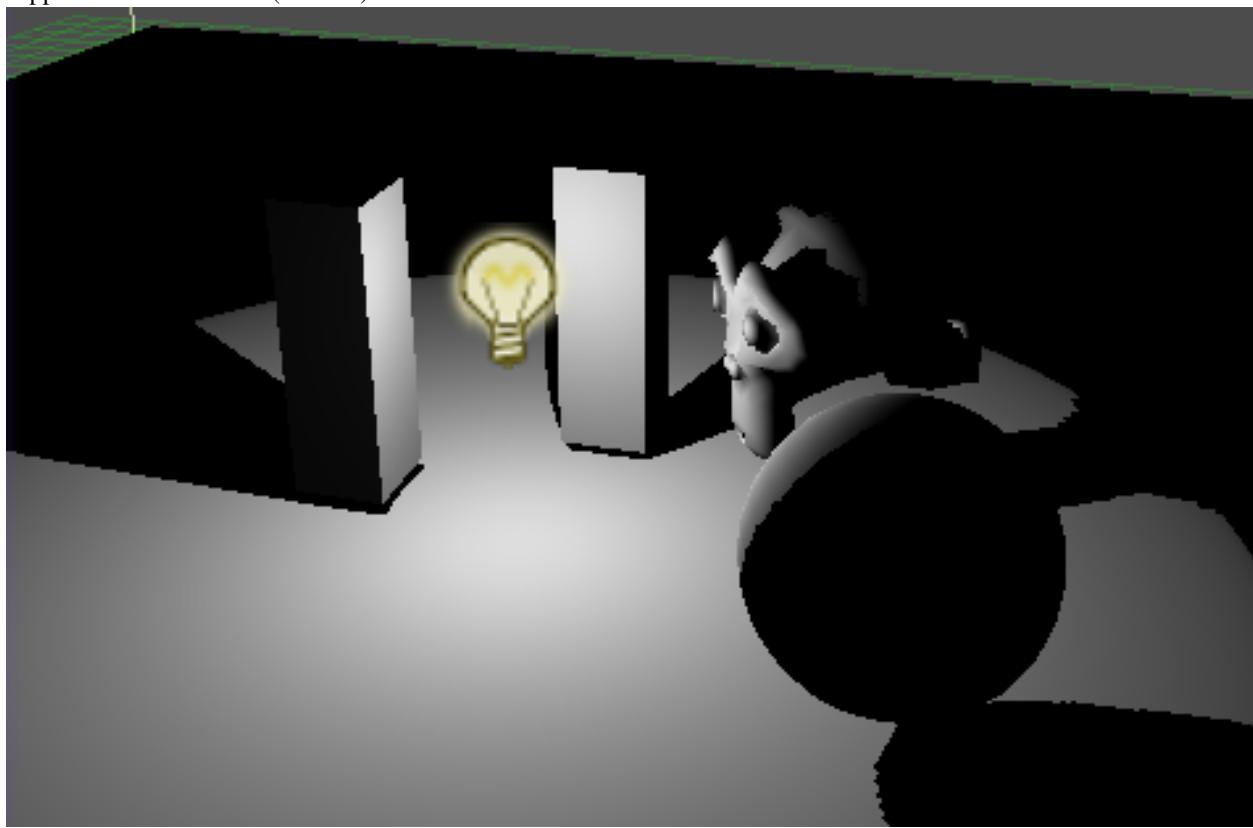


These techniques divide the view in 2 or 4 sections, and a shadow is rendered for each. This way, close objects can use larger shadow while further away objects will use one in less detail, but in proportion this seems to make the shadow map size increase while it's actually kept the same. Of course, this technique is not free, the more splits the more the performance goes down. On mobile, generally it is convenient to not 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).

### Omni Light Shadows

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.

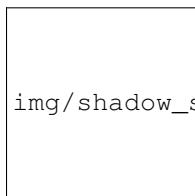


As few considerations when using DPSM shadow maps:

- Keep Slope-Scale on 0.
- Use a small value for Z-Offset, if things 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.



img/shadow\_spot.png

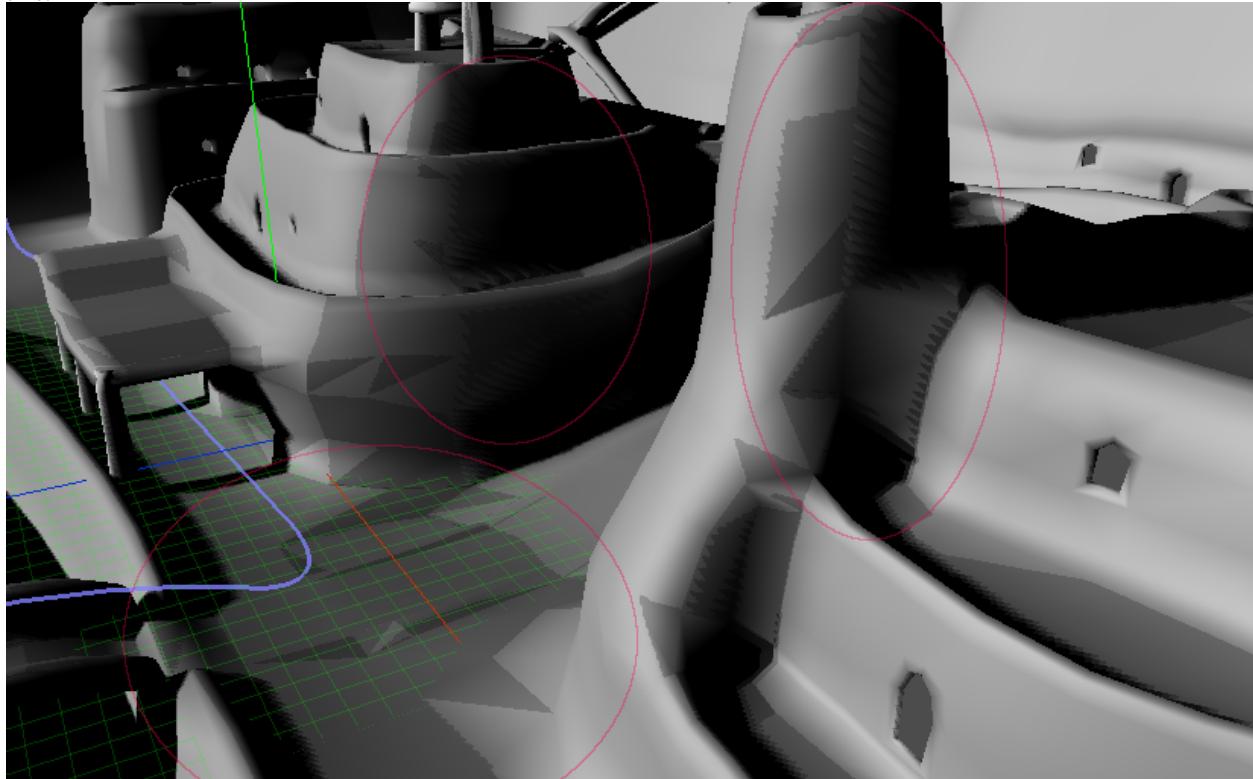
## Shadows Parameters

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.

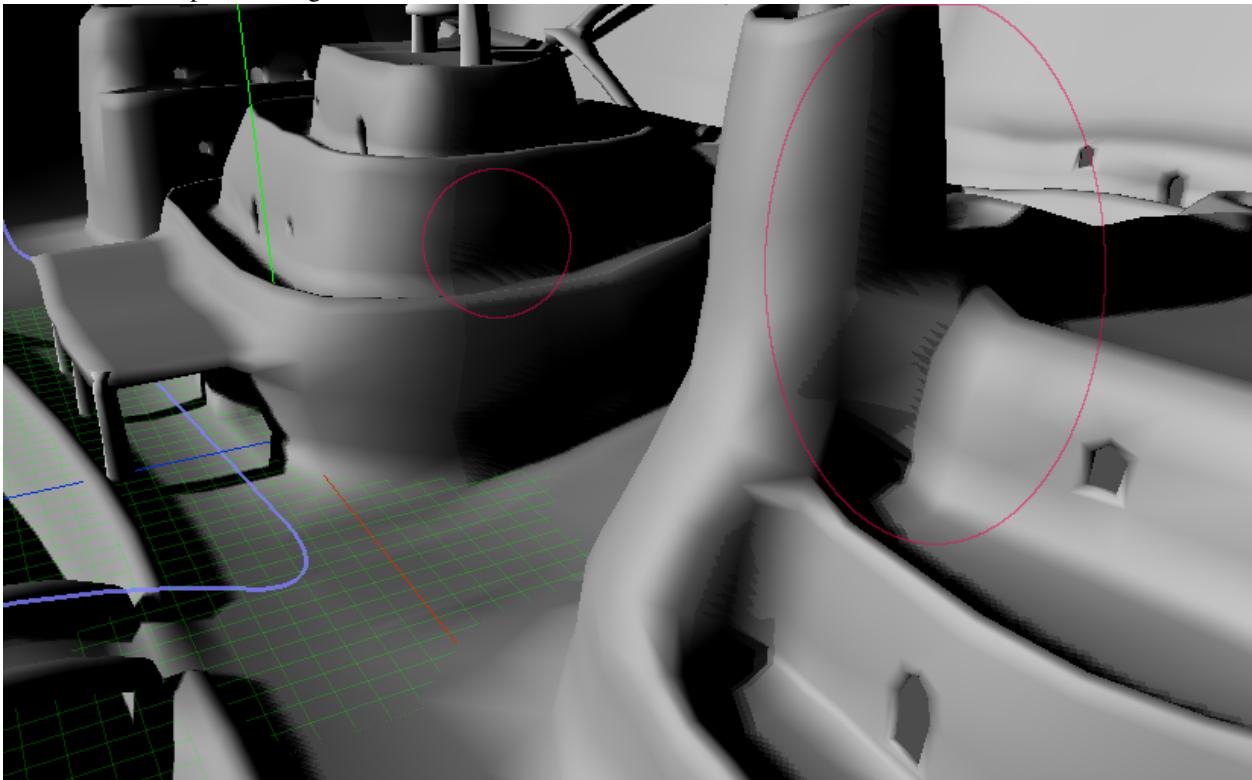
### Adjusting Z-Offset

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 kind of be a pain, but most of the time it can be solved to 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.

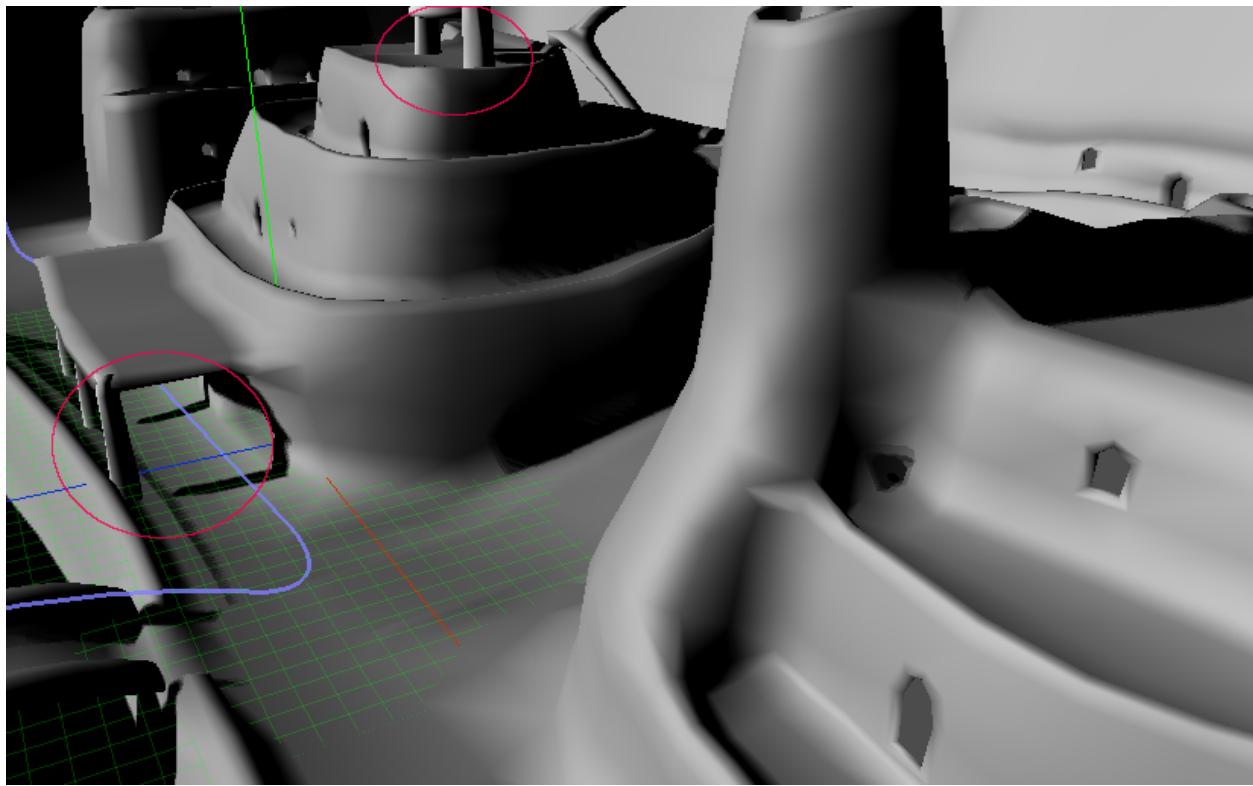
First step is to turn on the shadows, let's assume that both Z-Offset and Z-Slope-Scale are at 0. You will be greeted by this:



Holy crap, shadow is all around 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 adjusting the shadow Z-Offset. This improves things a lot:

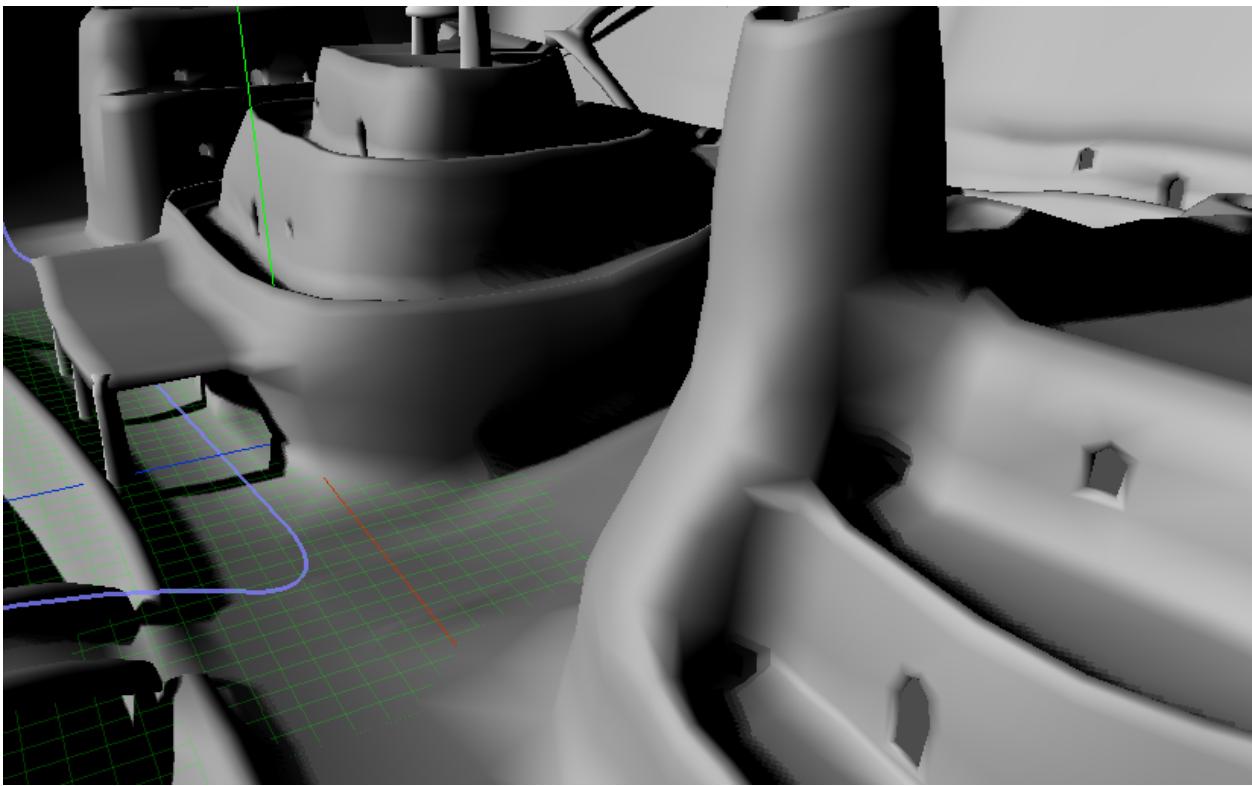


But it's not quite perfect, self shadowing did not dissapear completely. So close to perfection but still not there.. so in a turn of greed you increase the Z-Offset even more!

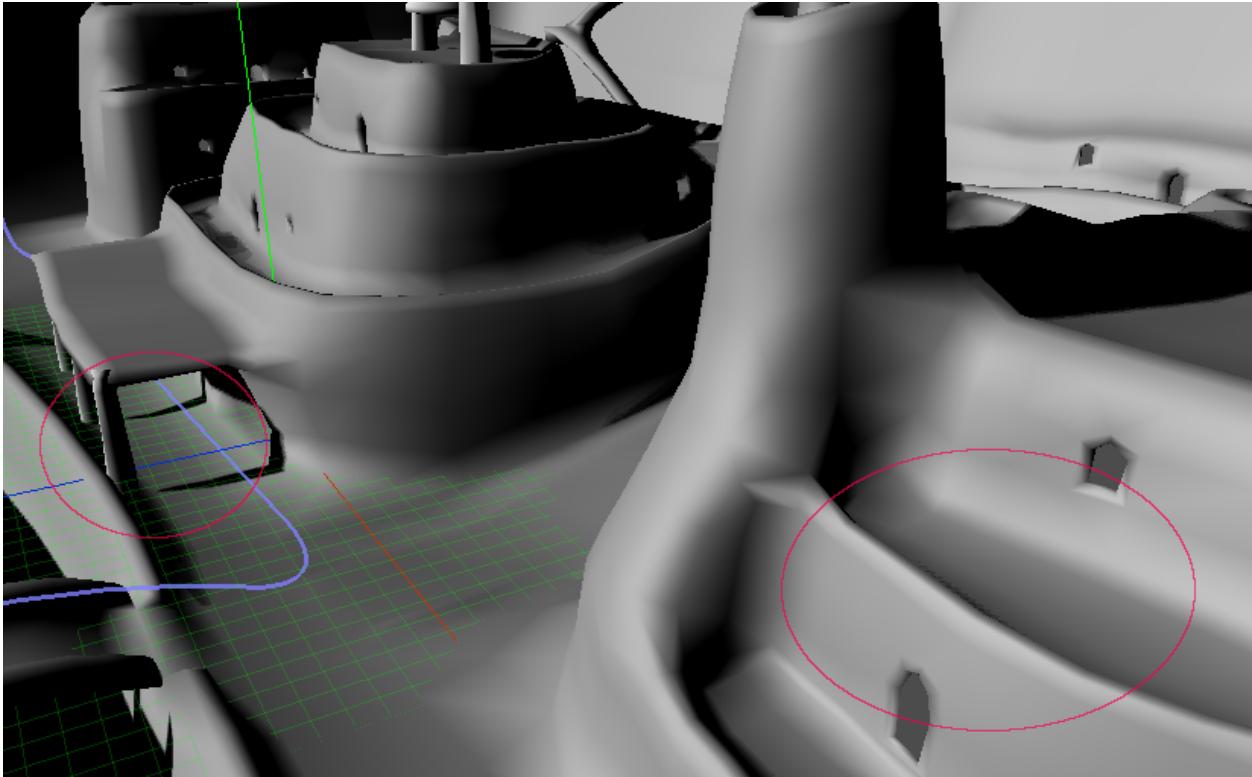


And it gets rid of those self-shadowings! hooray! except something is wrong.. oh, right. Being pushed back too much, the shadows start disconnecting from their casters, which looks pretty awful. Ok, you go back to the previous Z-offset.

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 horribile attempt at 3D modelling). There may be very tiiny 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!**

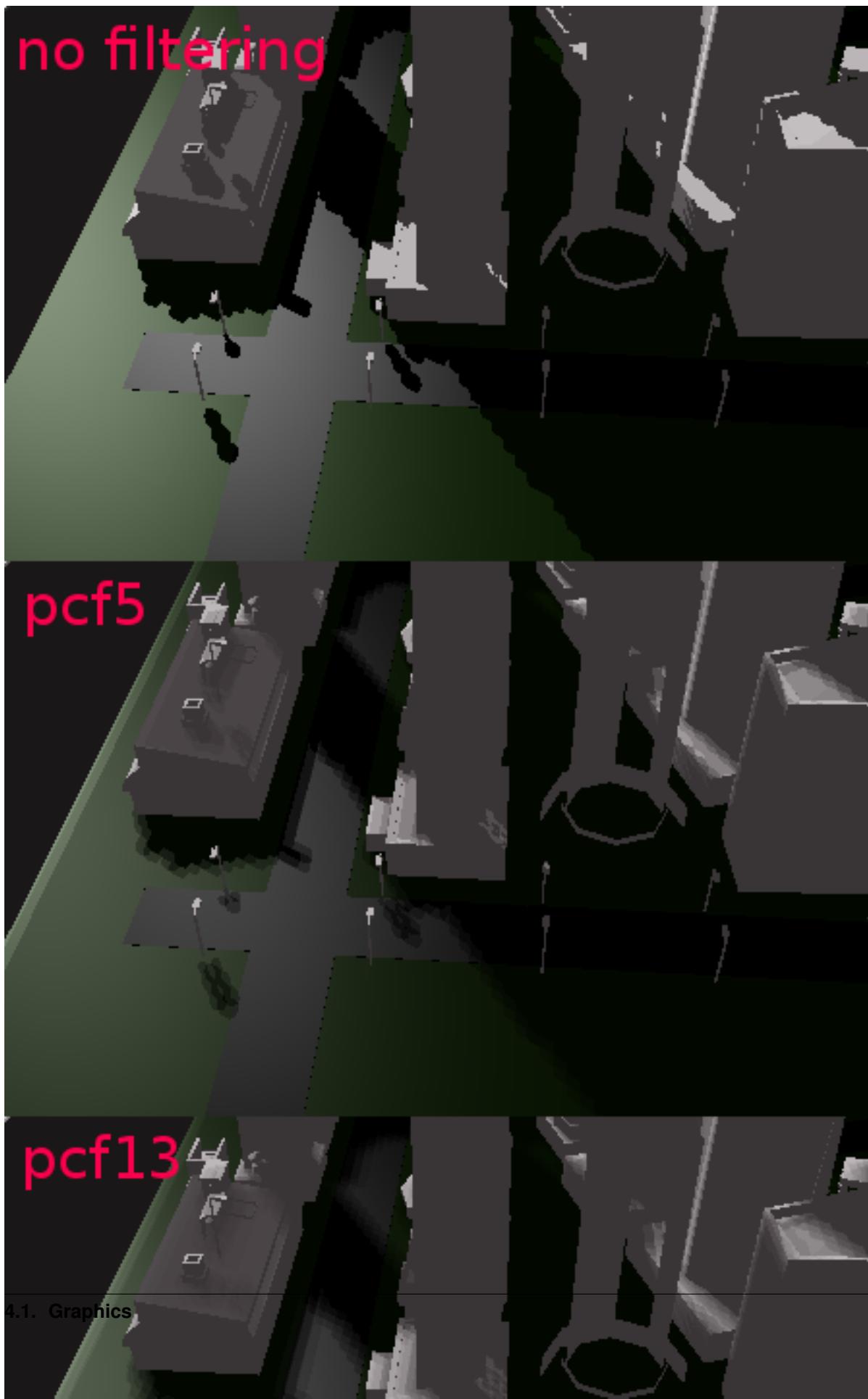
If you are using shadow maps with directional lights, make sure that the *view distance* of the *camera* is set to an *optimal range*. This means, if the distance between your camera and the visible end of the scene is 100, then set the view distance to that value. If a greater than necessary value is 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!):



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).

## 4.1.7 High Dynamic Range

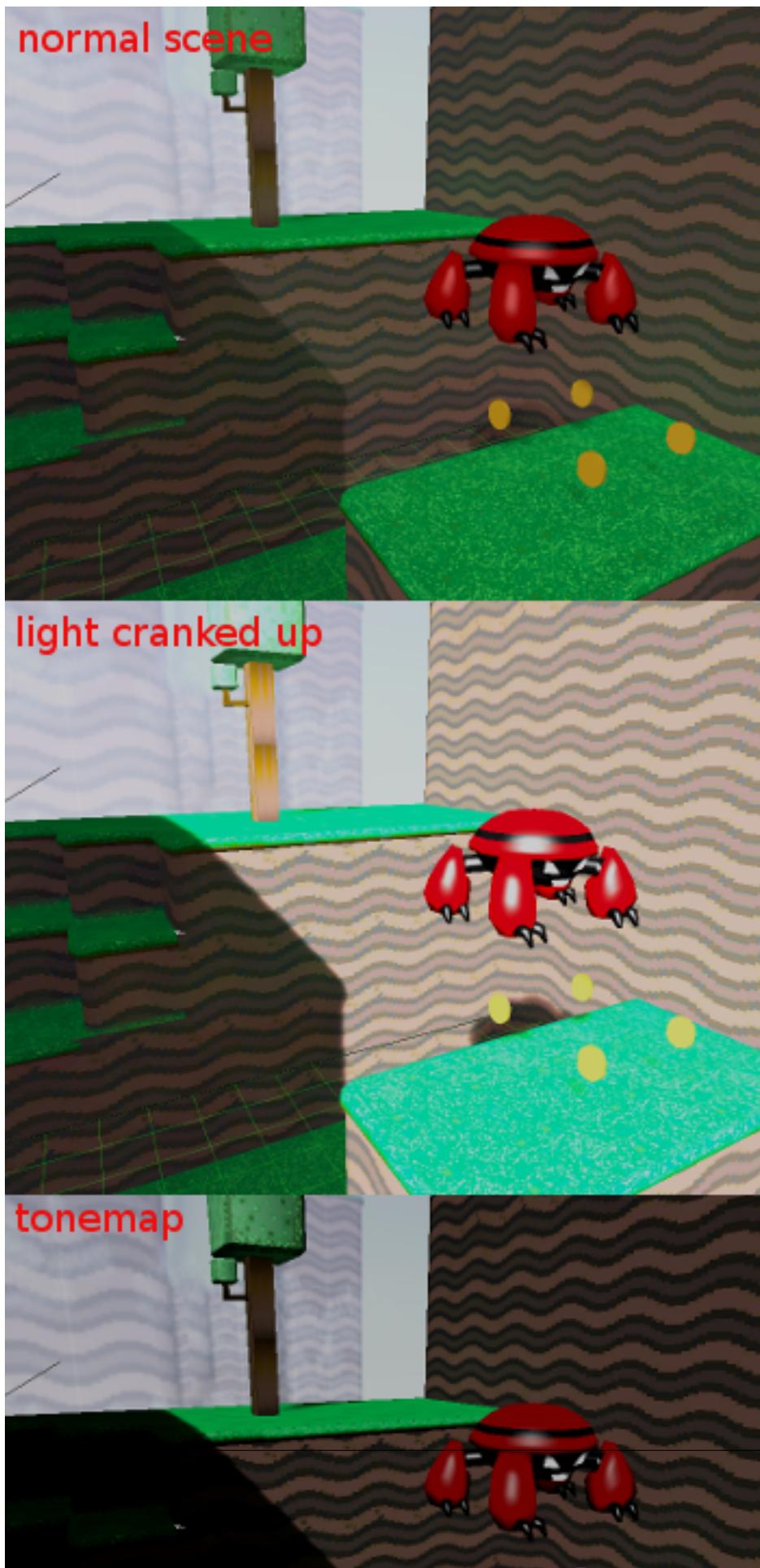
### Introduction

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.

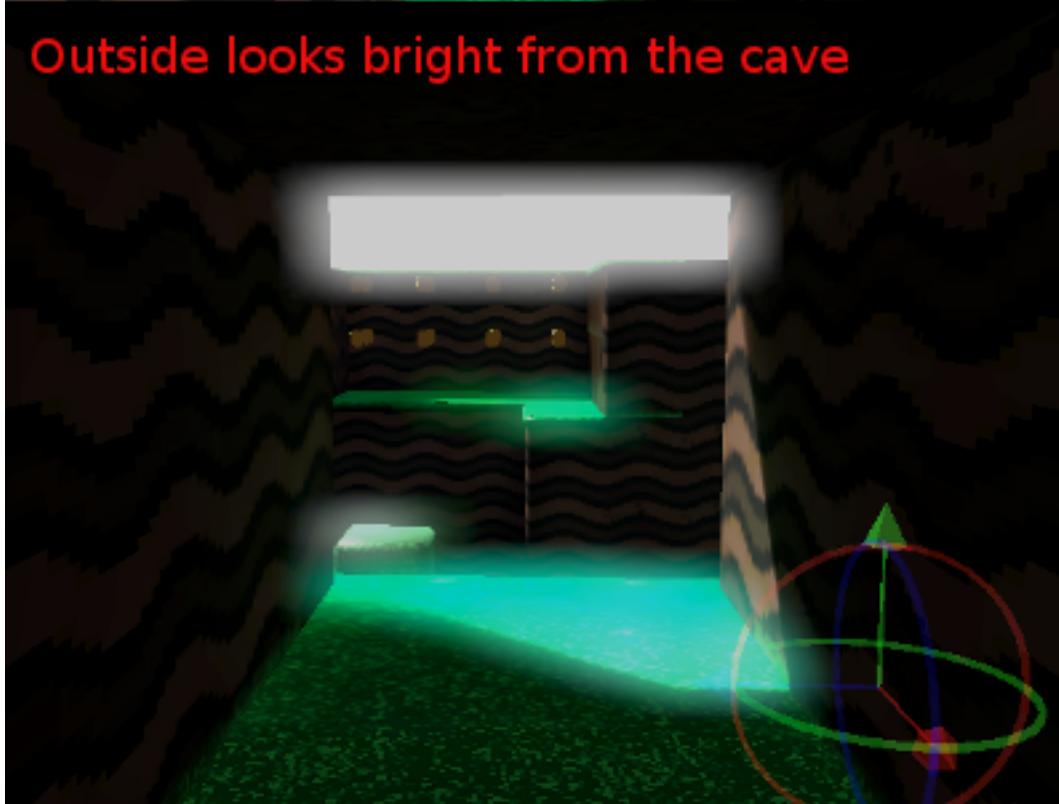
After this the luminance of the scene is computed by averaging the luminance of every pixel of it, and this value is used to bring the scene back to normal ranges. This last operation is called tone-mapping. Finally, we are at a similar place from where we started:



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:



Outside looks bright from the cave

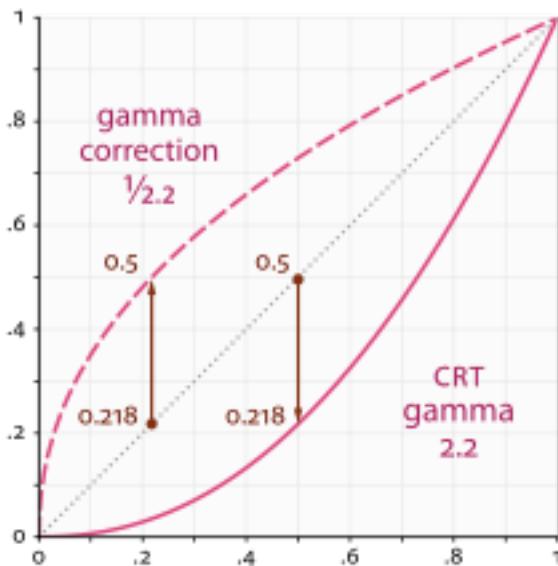


Additionally, it is possible to set a threshold value to send to the glow buffer depending on the pixel luminance. This allows for more realistic light bleeding effects in the scene.

## 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”. Every visual content that people has on their computers or downloads from the internet (such as pictures, movies, porn, 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.

## 4.1.8 3D Performance & 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 less objects are rendered, so efficiency overall improves.
- Configuring the properties of every material for every object that needs to be renderer 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).
1. 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 ([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.

### 4.1.9 Working with 3D skeletons

Godot 3D skeleton support is currently quite rudimentary. Skeleton node and class were designed mainly to support importing skeletal animations as 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 but not documented.

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

```
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**

```
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**

```
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

```
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

```
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 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:

```
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

```

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)

```

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:

```

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 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 way to change them. Just lets 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
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:

```
extends 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"):
        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

```
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

```
{{include("Inverse Kinematics")}}
```

### Using 3D “bones” to implement ragdoll-like physics

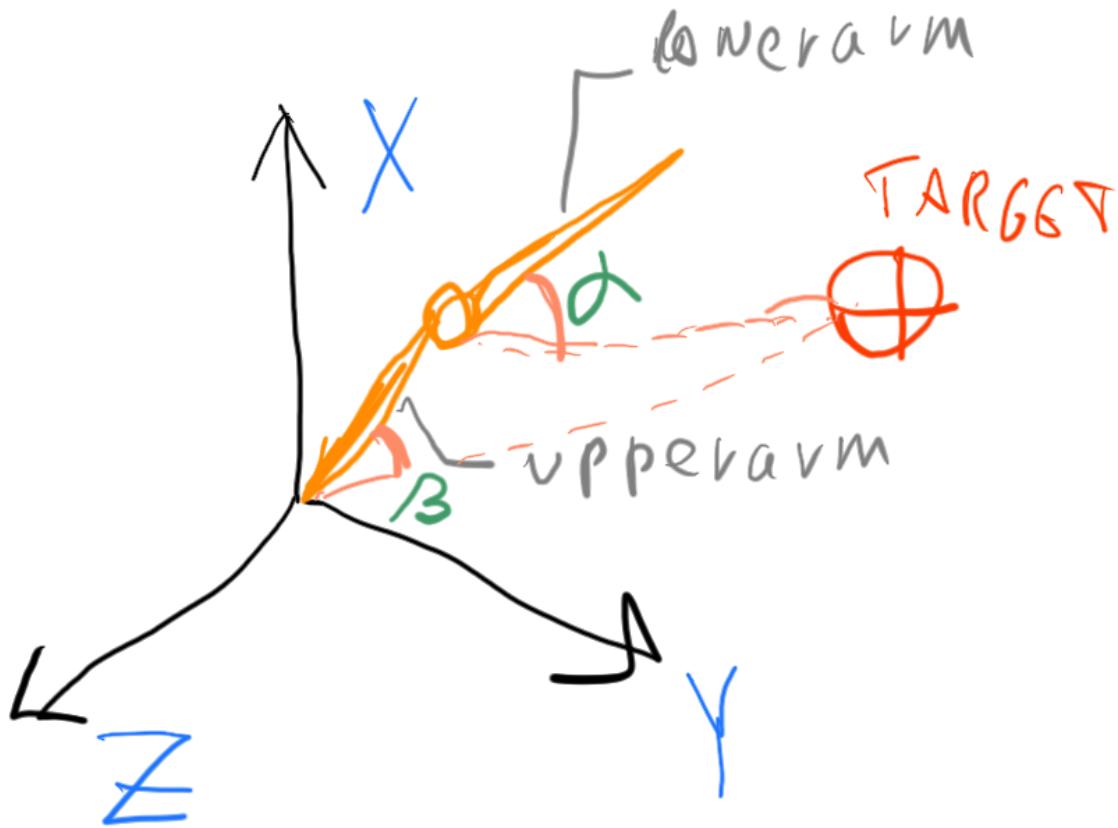
TBD

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](http://freespace.virgin.net/hugo.elias/models/m_ik2.htm)

#### 4.1.10 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. Iso, it is 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 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 tip of the bone?**
- **how to find vector from bone base to target?**

For our goal (tip of the bone is within area of target), we need to know where is a tip of our IK bone. As we don't use leaf bone as IK bone, we know, that coordinate of tip is the base of child bone. But all these calculations are quite depend on skeleton structure. You can use pre-calculated constant as well. You can add extra bone for the tip of IK and calculate using that.

### 4.1.11 Implementation

We will just use exported variable for bone length to be easy.

```
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.

```
export var IK_limit = 2
```

For `_ready()` function:

```
var skel
func _ready():
    skel = get_node("arm/Armature/Skeleton")
    set_process(true)
```

Now we can write our chain-passing function:

```
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:

```
func _process(dt):
    pass_chain(dt)
```

Executing this script will just pass through bone chain printing bone transforms.

```
extends Spatial

export var IK_bone="lowerarm"
export var IK_bone_length=1.0
export var IK_error = 0.1
export var IK_limit = 2
var skel
func _ready():
    skel = get_node("arm/Armature/Skeleton")
    set_process(true)
func pass_chain(dt):
    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
```

```
func _process(dt):
    pass_chain(dt)
```

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:

```
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)
```

## 4.2 Physics

## 4.3 Import

### 4.3.1 Importing 3D meshes

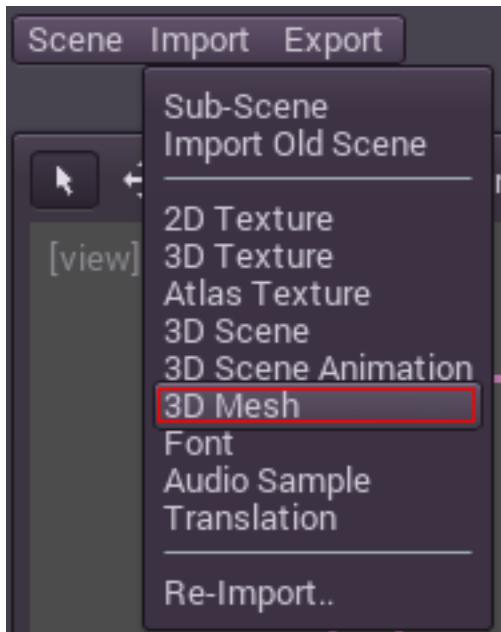
#### Introduction

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).

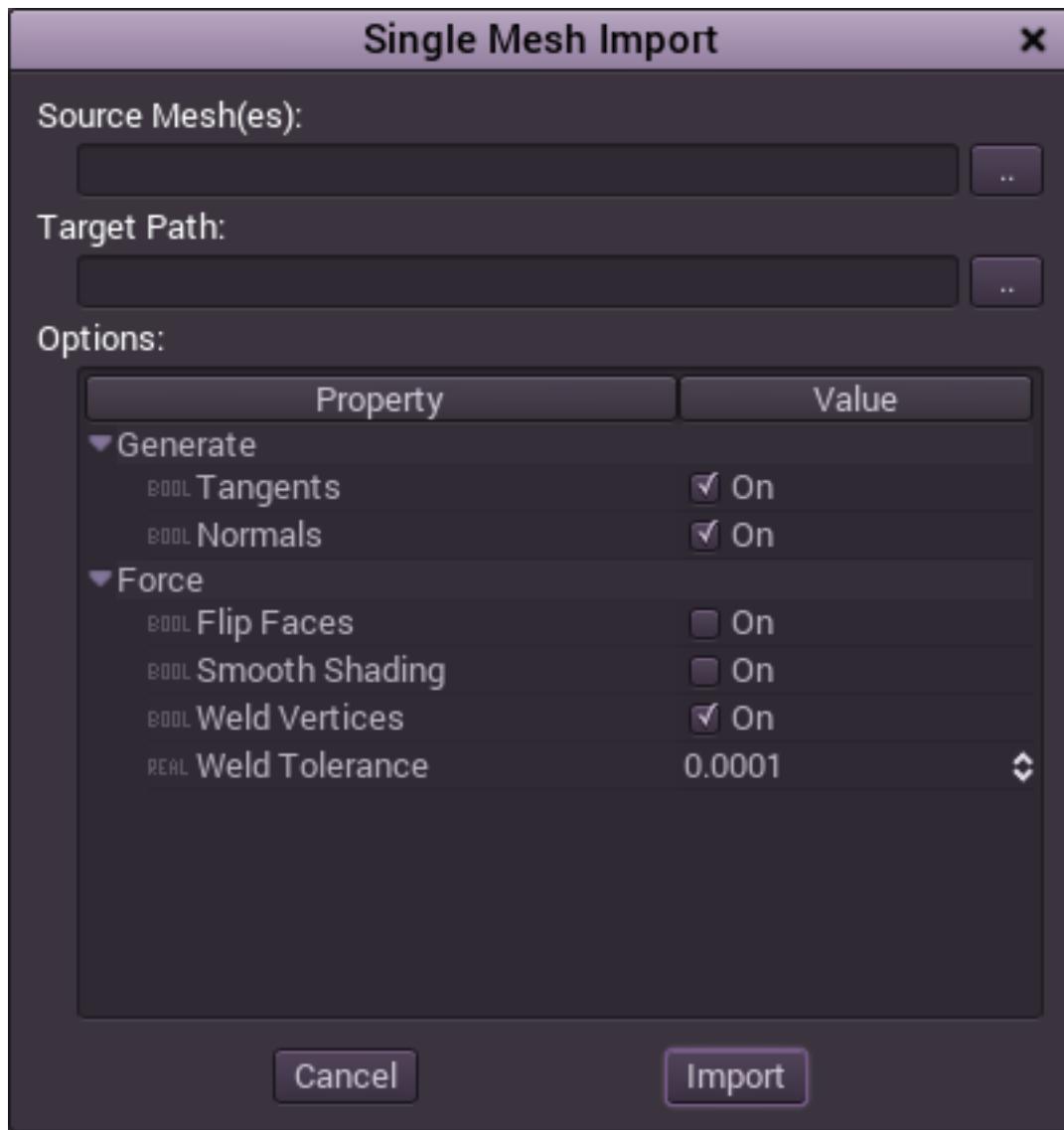
The 3D mesh import workflow is simple and works using the OBJ file format. The imported meshes result in a .msh binary file which the user can put into a [[API:MeshInstance]], which in turn can be placed somewhere in the edited scene.

#### Importing

Importing is done through the Import 3D Mesh menu:



Which opens the Mesh import window:



This dialog allows the import of one or 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 is needed for regular shading, while Tangents is 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) are used inside MeshInstance nodes. Simply set them to the Mesh property of them.



And that is it.

### 4.3.2 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 (Such as Maya, 3DS Max or Blender) tools 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

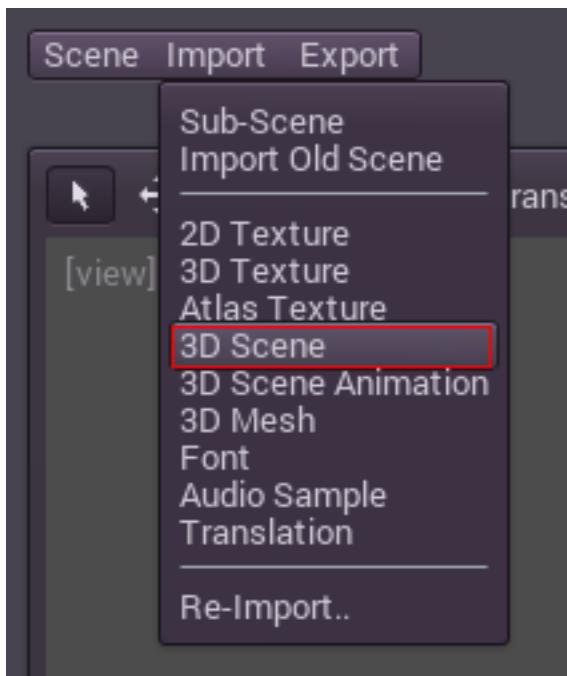
Autodesk added built-in collada support to Maya and 3DS Max, but It's really broken and should not be used. The best way to export this format is by using the [OpenCollada](#) plugins. They work really well, although they are not always up-to date with the latest version of the software.

### 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:

p=. [image1](#)

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 (.dae stands for Collada. 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, pvr, 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 you keep them outside the project. For some hints on how to do this the best way, you can check the [[Version control & 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 names 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

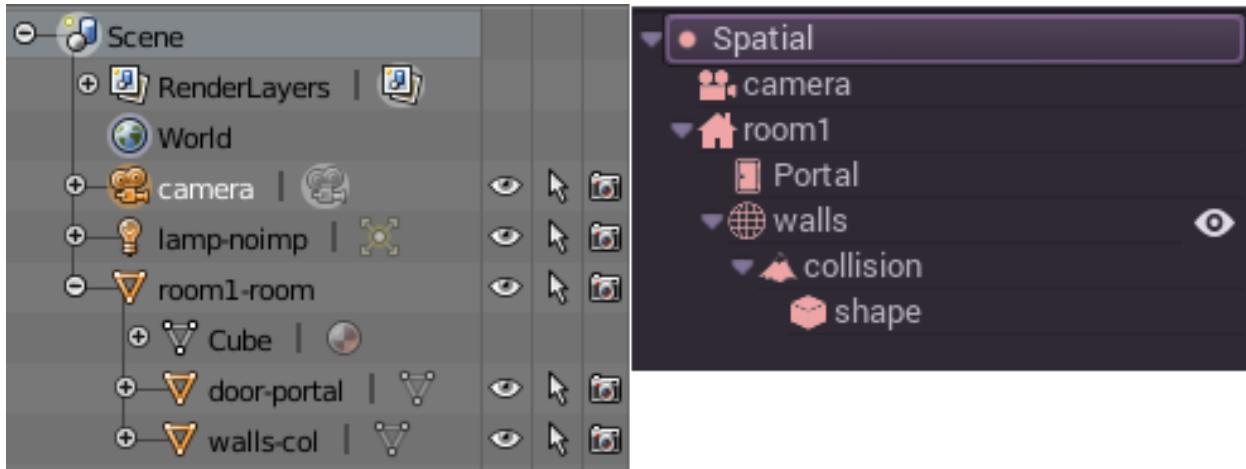
This section contains many options to change the way import workflow works. Some (like HDR) will be better explained in other sections, but in general a pattern can be visible in the options and that is, many of the options end with “-something”. For example:

- 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 they will be restored if the source scene changes.

**Import animations** Some scene formats (.dae) support one or more animations. If this is checked, an Animation-Player 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 [[HDR]].

**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 apps 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 [Fixed Ma

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)** These will only work for Mesh nodes, If the “-col” option 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.

**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). For more information about rooms/portals, look at the [[Portals and Rooms]] tutorial.

There are two ways in which this modifier can be used. The first is using a Dummy/Empty node in the 3D app with the “-room” tag. For this to work, the “interior” of the room must be closed (geometry of the childrens 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..).

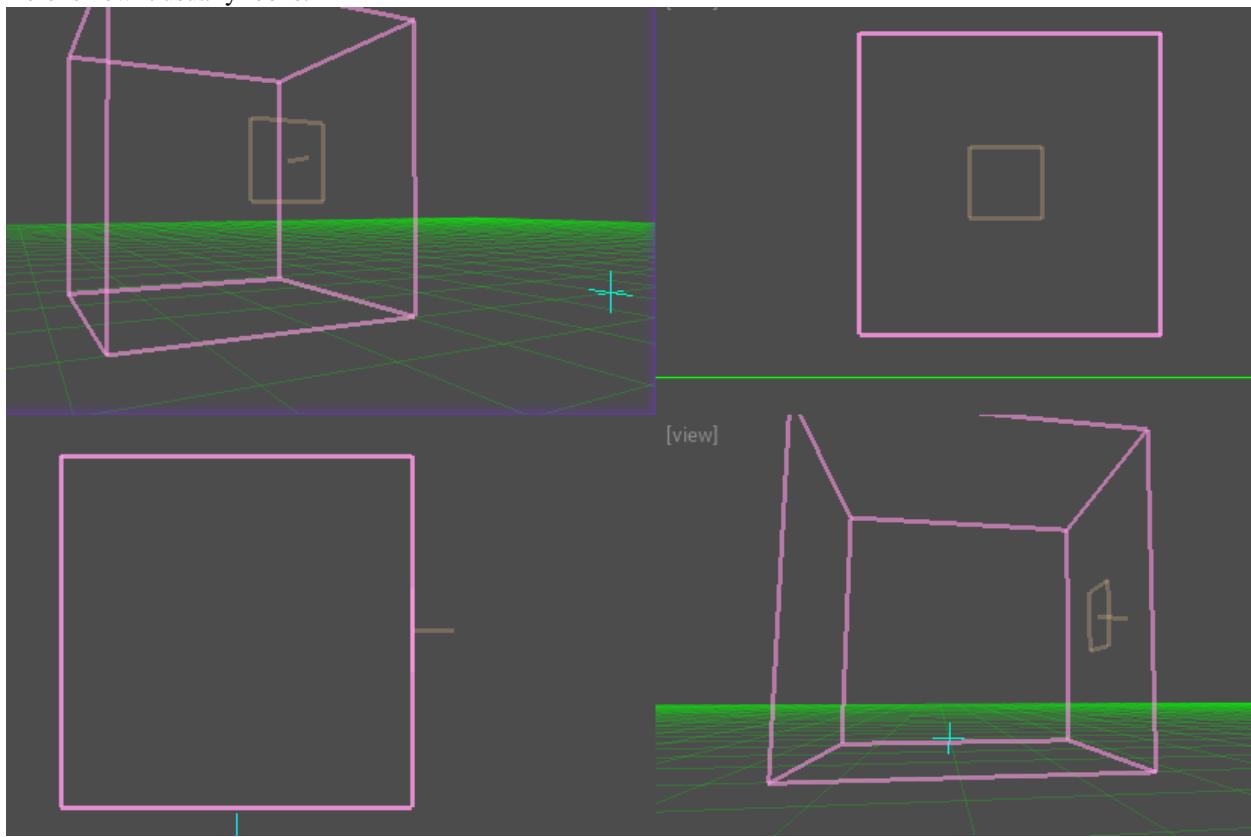
Anyway, the room will need portals, which are described next.

**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. Again, more information on the [[Portals and Rooms]] tutorial.

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 ok, 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 or 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 get to break an animation, so make it sure to disable if you see this.

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:

```
tool #needed so it runs in editor
extends EditorScenePostImport

func post_import(scene):
    #do your stuff here
    pass # scene contains the imported scene starting from the root node
```

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.



---

## **Networking**

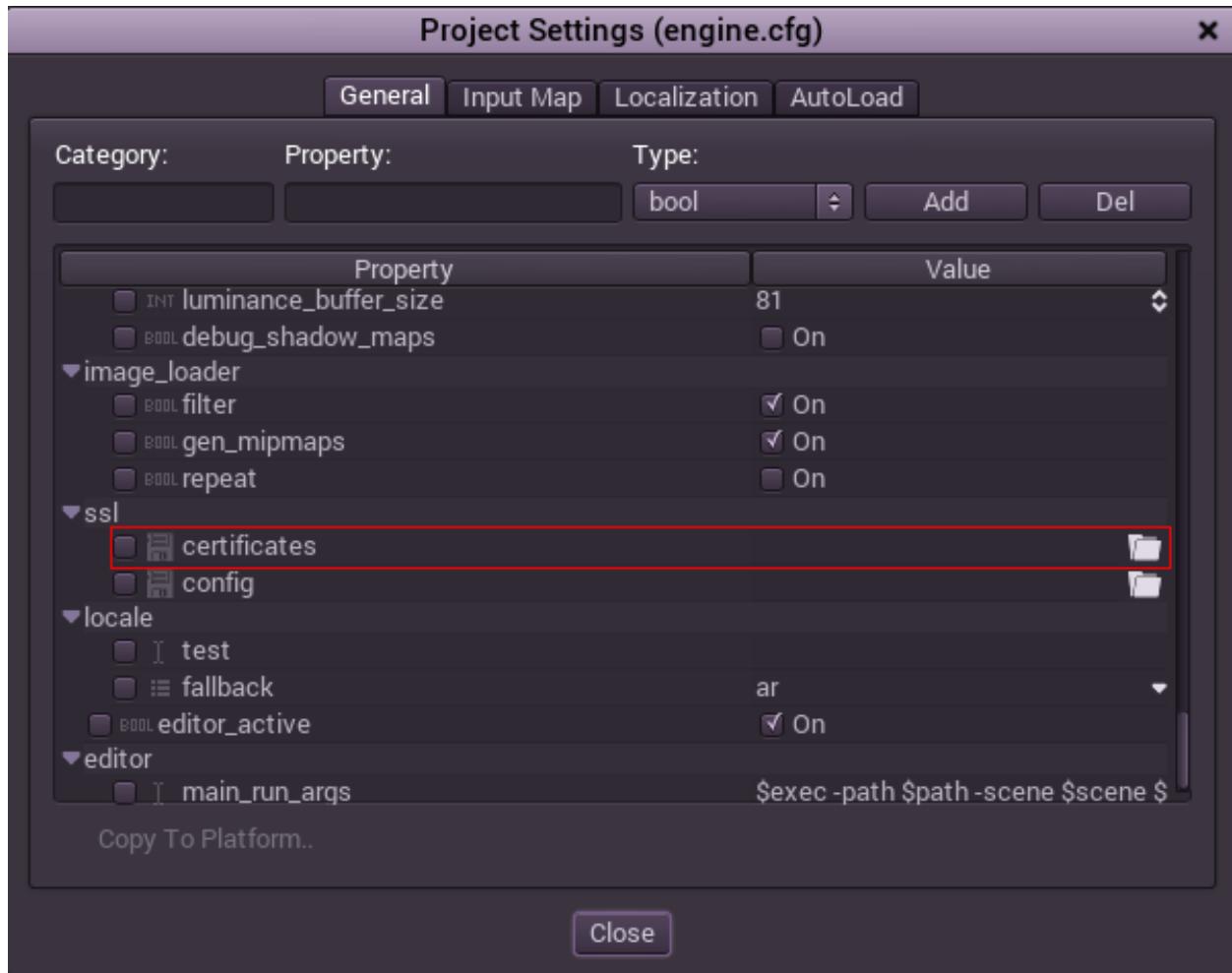
---

### **5.1 SSL Certificates**

#### **5.1.1 Introduction**

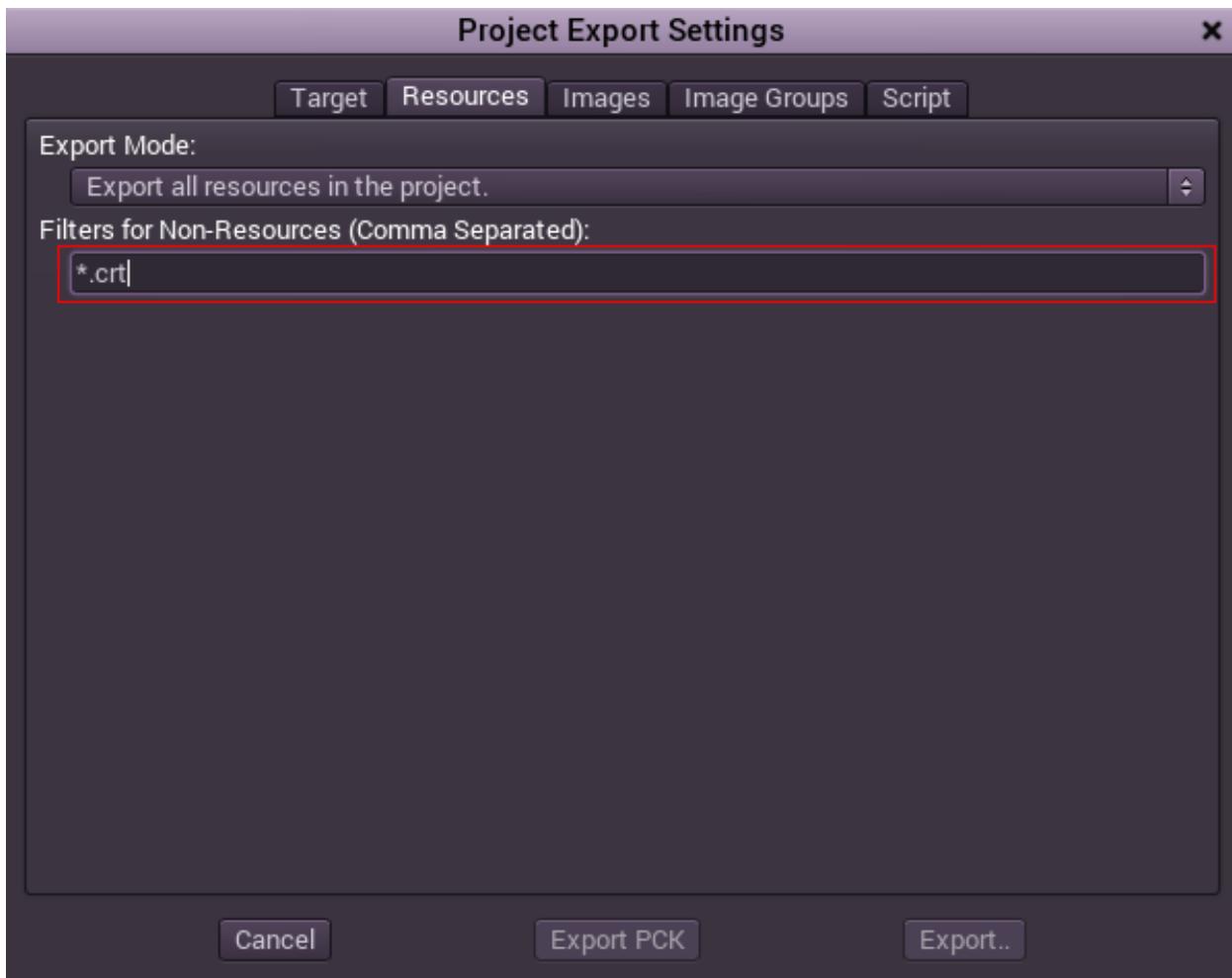
It is often desired to use SSL connections for communications to avoid “man in the middle” attacks. Godot has a connection wrapper, [StreamPeerSSL](#), which can take a regular connection and add security around it. The [HTTPClient](#) class also supports HTTPS by using this same wrapper.

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](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:

### 5.1.2 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.

### 5.1.3 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 (ie, 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 (ie, Google, Microsoft, etc) .

Or just pick any of the more specific certificates there if you are connecting to a specific one.

## 5.2 HTTP client class example

Here's an example of using the `HTTPClient` class. It's just a script, so it can be run by executing:

```
c  
c:\\godot> godot -s http_test.gd
```

It will connect and fetch a website.

```
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

    while( http.get_status()==HTTPClient.STATUS_CONNECTING or http.get_status()==HTTPClient.STATUS_REQUESTING):
        #Wait until resolved and connected
        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 page from the server

    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)
```

```
assert( http.get_status() == HTTPClient.STATUS_BODY or http.get_status() == HTTPClient.STATUS_CODE)

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("Respose 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 bufer

    #done!

    print("bytes got: ",rb.size())
    var text = rb.get_string_from_ascii()
    print("Text: ",text)

quit()
```



## **Editor plugins**

---

Coming soon™.



---

## Miscellaneous

---

# 7.1 Math

## 7.1.1 Vector Math

### Introduction

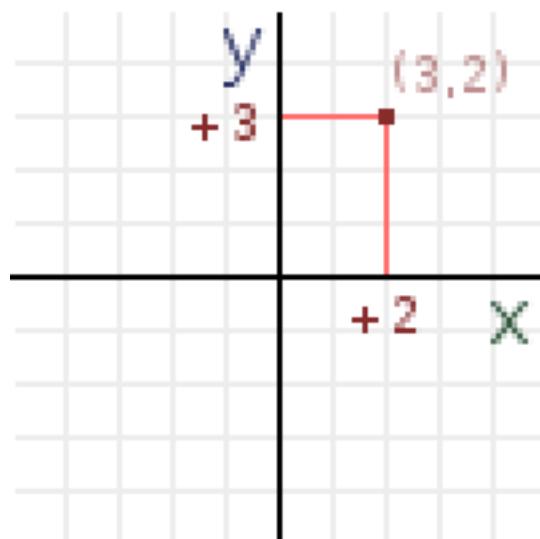
This small tutorial aims to be a short and practical introduction to vector math, useful for 3D but also 2D games. Again, vector math is not only useful for 3D but *also* 2D games. It is an amazing tool once you get the grasp of it and makes programming of complex behaviors much simpler.

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 of math for 2D games.

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

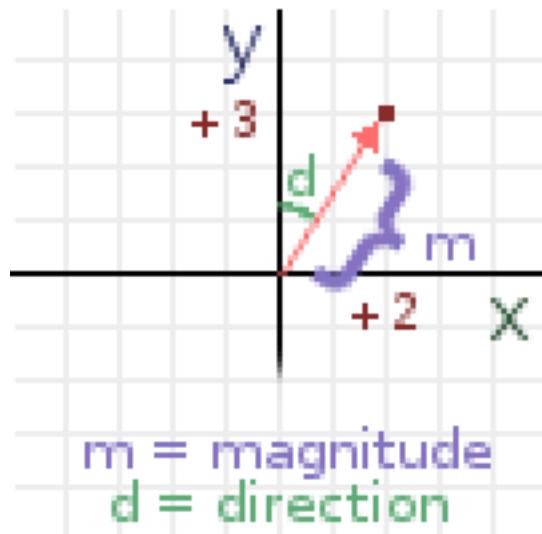
### Coordinate Systems (2D)

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.

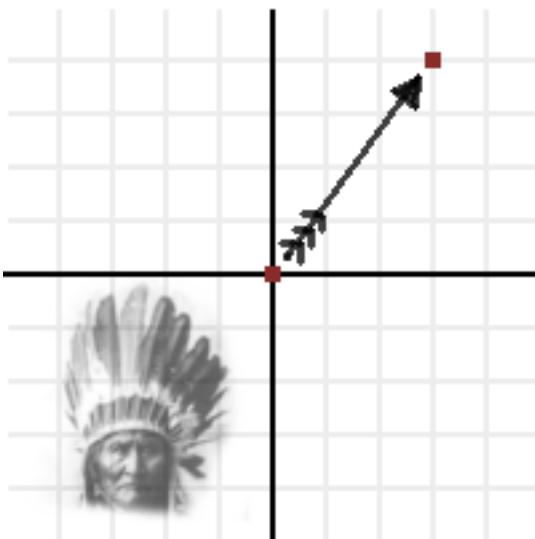


When imagined as a vector, two properties can be inferred, the **direction** and the **magnitude**. Every position in space can be a vector, with the exception of the **origin**. This is because coordinates (0,0) can't represent direction (magnitude 0).



### Direction

Direction is simply towards where the vector points to. Imagine an arrow that starts at the **origin** and goes towards a [STRIKEOUT: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 [Pithagorean Theorem](#).

```
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 atan2(x,y) function.`

## Vectors in Godot

To make examples easier, it is worth explaining how vectors are implemented in GDScript. GDscript has both `Vector2` and `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.

```
h1. create a vector with coordinates (2, 5)
var a = Vector2(2, 5)
h1. 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:

```
#add a and b
var c = a+b
h1. will result in c vector, with value (9,13)
```

It is the same as doing:

```
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 **scalars**.

```
h1. Multiplication of vector by scalar
var c = a*2.0
h1. will result in c vector, with value (4,10)
```

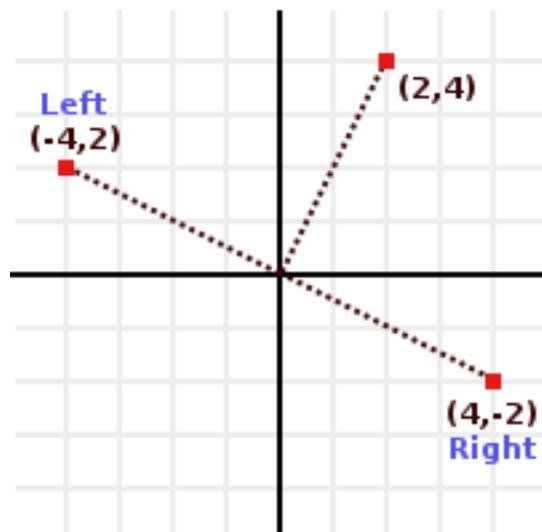
Which is the same as doing

```
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).



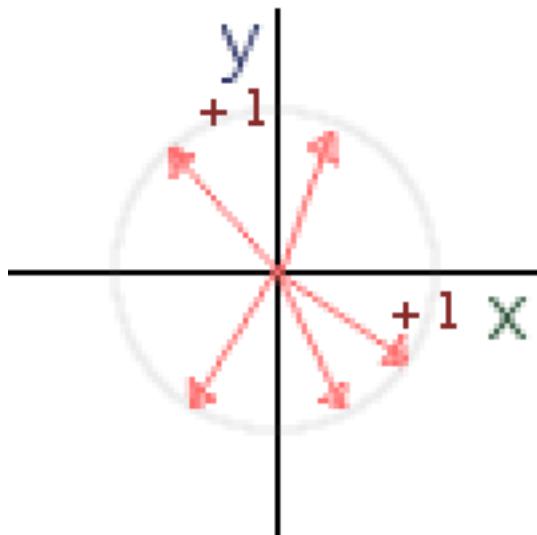
Example:

```
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 it's **magnitude** to 1.0 while keeping it's **direction** is called **normalization**. Normalization is performed by dividing the x and y (and z in 3D) components of a vector by it's magnitude:

```
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:

```
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**:

```
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:

```
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:

```
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:

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

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 well 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* why do they exist for.

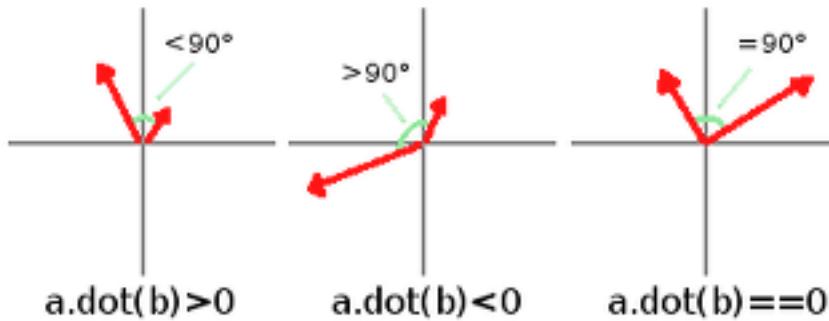
## Siding

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.

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

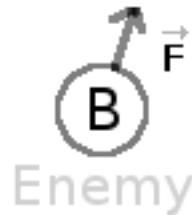
The operation will return a single floating point number (but since we are in vector world, we call them **scalar**, will keep using that term from now on). This number will tell us the following:

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



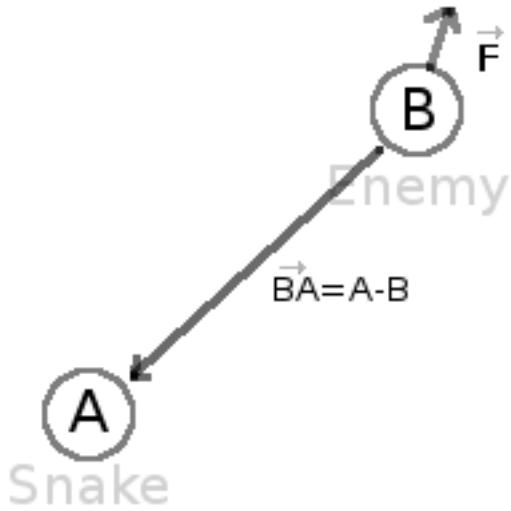
So let's think of a real use-case scenario. Imagine Snake is going through a forest, and then there is an enemy nearby. How can we quickly tell if the enemy has seen discovered Snake? In order to discover him, the enemy must be able to *see* Snake. Let's say, then that:

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



So, let's create a new vector **BA** that goes from the guard (**B**) to Snake (**A**), by subtracting the two:

```
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 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:

```
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:

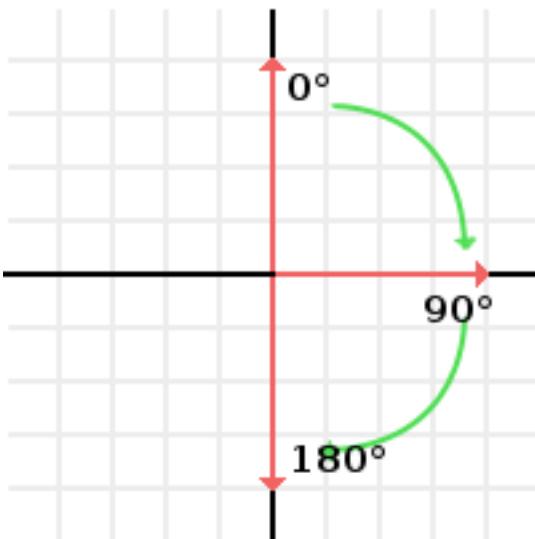
- If both vectors are facing towards the exact same direction (parallel to each other, angle between them is  $0^\circ$ ), the resulting scalar is **1**.
- If both vectors are facing towards the exact opposite direction (parallel to each other, but angle between them is  $180^\circ$ ), the resulting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

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

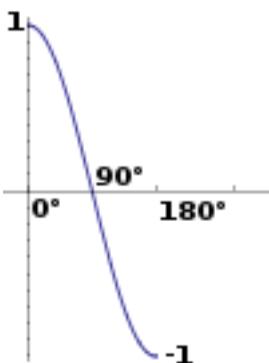
- If their angle is  **$0^\circ$**  dot product is **1**.
- If their angle is  **$90^\circ$** , then dot product is **0**.
- If their angle is  **$180^\circ$** , then dot product is **-1**.

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 ( $0^\circ$ ) to down ( $180^\circ$  degrees)..



..while plotting the resulting scalar!



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) )
```

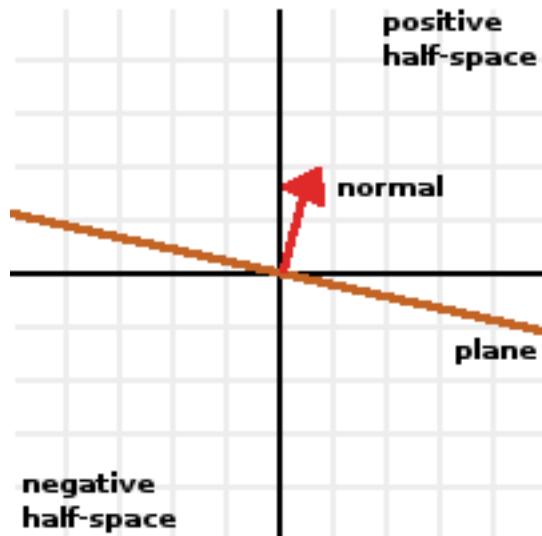
What is this useful for? Well obtaining the angle directly is probably not as useful, but just being able to tell the angle is useful for reference. One example is in the [Kinematic Character](#) demo, when the character moves in a certain direction then we hit an object. How to tell if what we hit is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



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)!).

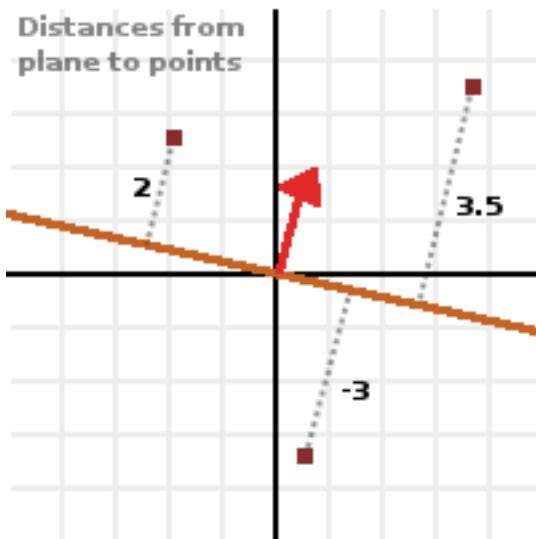
It's as simple as it looks. The plane passes by the origin and the surface of it is perpendicular to the unit vector (or *normal*). The side towards the vector points to is the positive half-space, while the other side is the negative half-space. In 3D this is exactly the same, except that the plane 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**:

```
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:



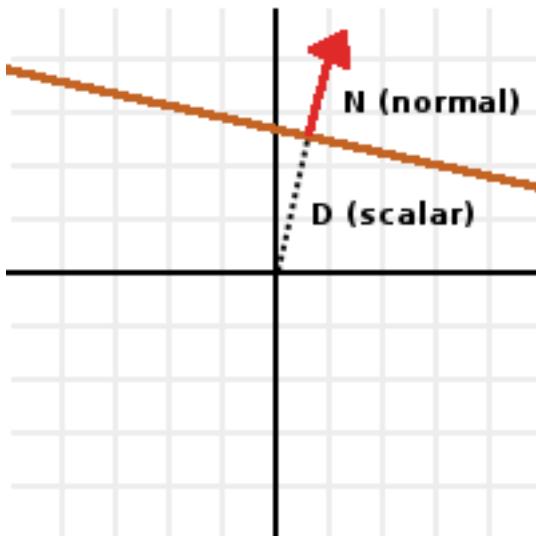
This allows us to tell which side of the plane a point is.

### #h3. 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:

```
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:

```
var distance = N.dot(point) - D
```

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:

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

Of course, Godot implements this operator in [Plane](#), so doing:

```
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.

```
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:

```
#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:

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

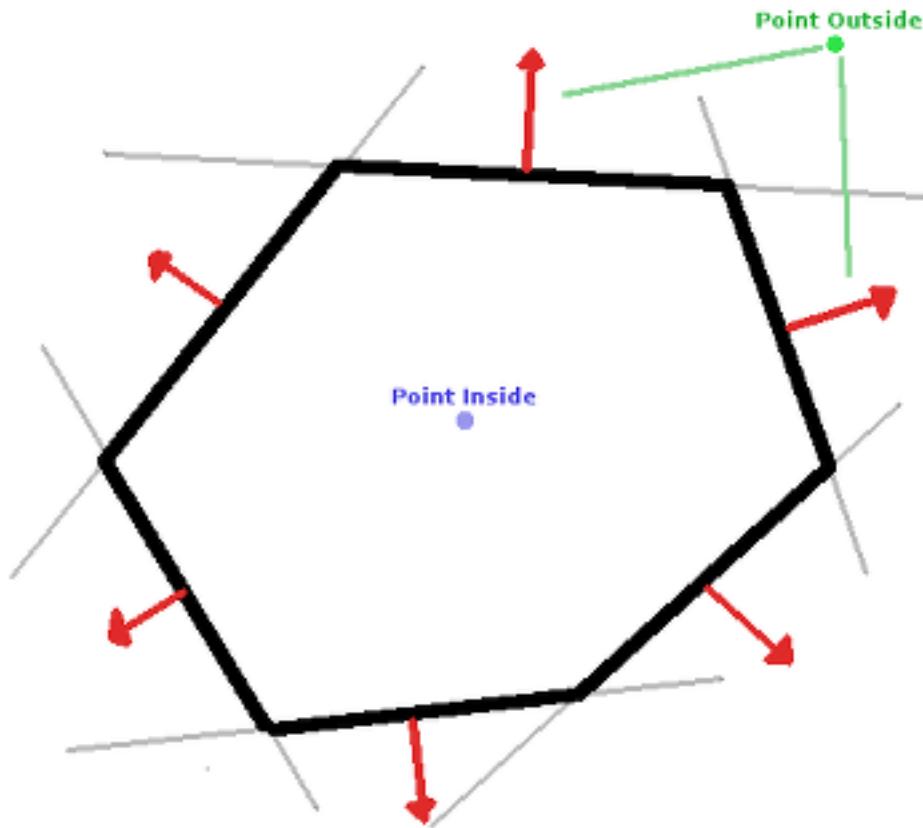
Doing the same in 3D is a little more complex and will be explained further down.

## 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.

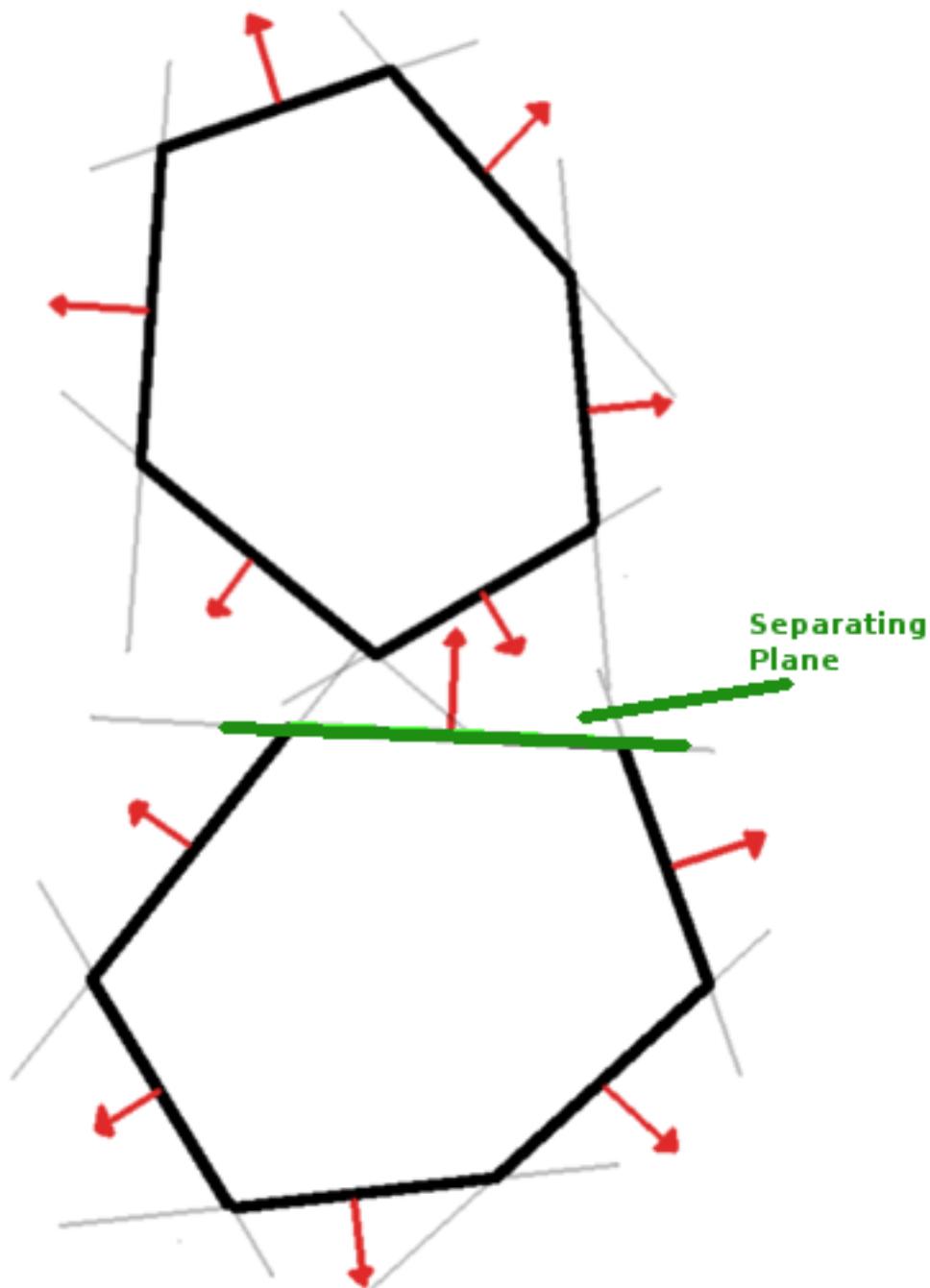


Code should be something like this:

```
var inside=true
for p in planes:
    #check if distance to plane is positive
    if ( N.dot(point) - D > 0):
        inside=false
    break h1. 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:

```
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:
        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).

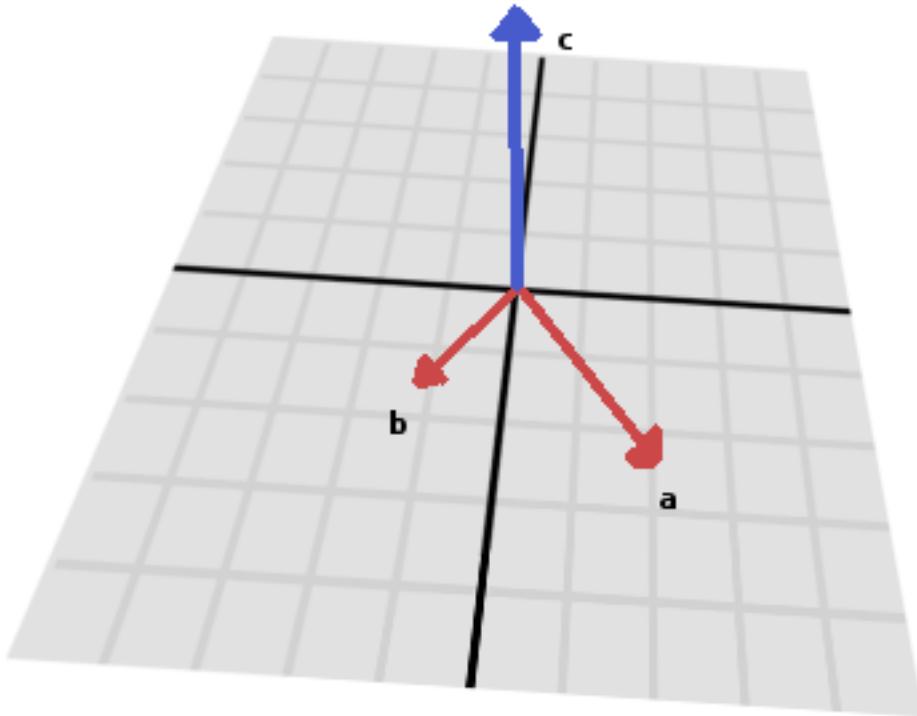
## Cross Product

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:

```
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:

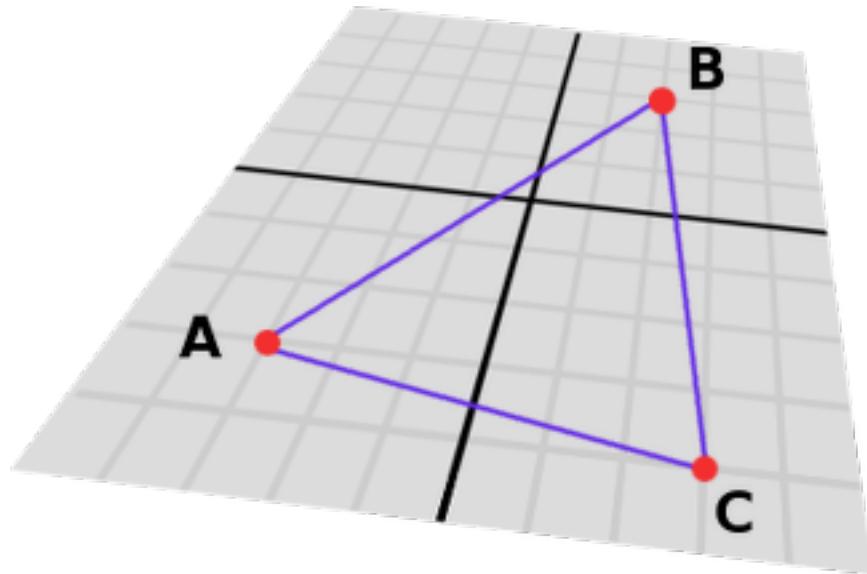
```
var c = a.cross(b)
```

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.

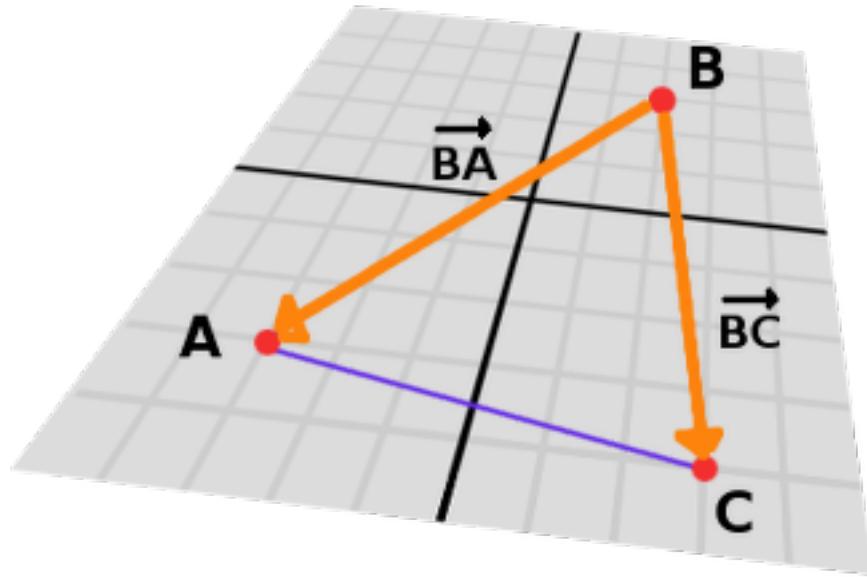
## Area of a Triangle

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**:



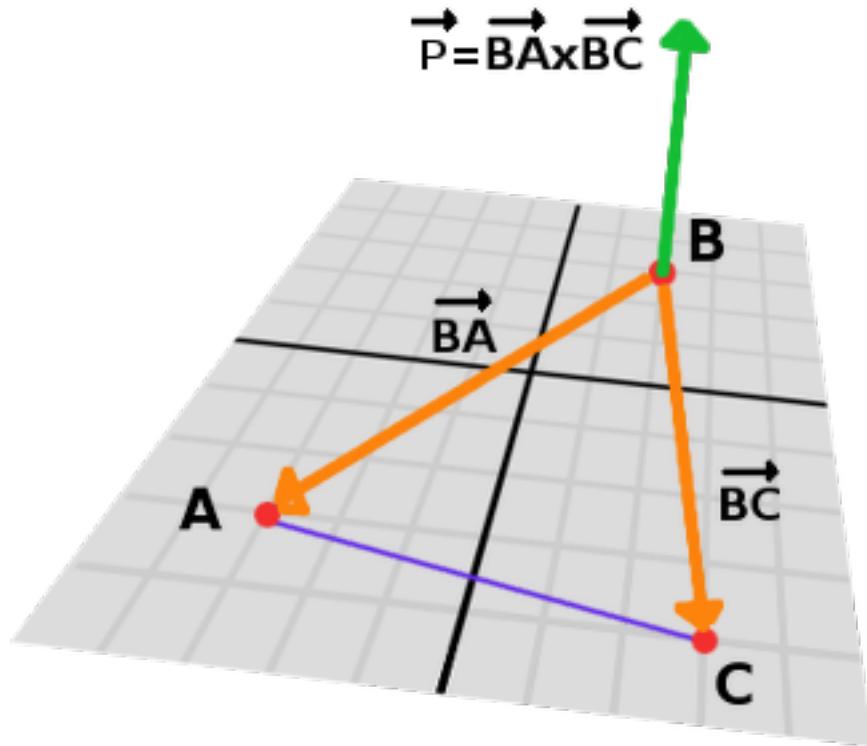
Take any of them as a pivot and compute the adjacent vectors to the other two points. As example, we will use **B** as a pivot:

```
var BA = A-B  
var BC = C-B
```



Compute the cross product between  $\overrightarrow{BA}$  and  $\overrightarrow{BC}$  to obtain the perpendicular vector  $\overrightarrow{P}$ :

```
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.

```
var area = P.length() / 2
```

## Plane of the Triangle

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

```
var N = P.normalized()
```

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

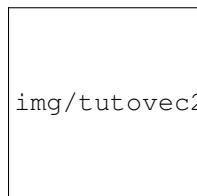
```
var D = P.dot(A)
```

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.

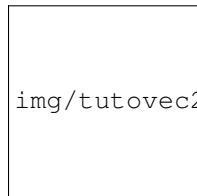
In Godot, front-facing triangles are those that, when looking at the camera, are in clockwise order. Triangles that look Counter-clockwise when looking at the camera are not drawn (this helps to draw less, so the back-part of the objects is not drawn).

To make it a little clearer, in the image below, the triangle **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.



img/tutovec20.png

Normals of triangles often are sided towards the direction they can be viewed from, so in this case, the normal of triangle ABC would point towards the front camera:



img/tutovec21.png

So, to obtain  $\mathbf{N}$ , the correct formula is:

```
# 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

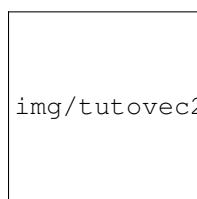
This is another bonus bit, a reward for being patient and keeping up with this long tutorial. Here is another piece of wisdom. This maybe is not something with a direct use case (Godot already does collision detection pretty well) but It's a really cool algorithm to understand anyway, because it's used 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.

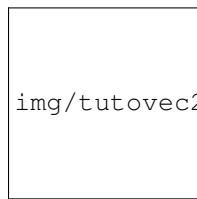
To refresh a bit a separating plane means that all vertices of polygon A are in one side of the plane, and all vertices of polygon B are in the other side. This plane is always one of the face-planes of either polygon A or polygon B.

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:



img/tutovec22.png

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



img/tutovec23.png

So the final algorithm is something like:

```
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:
        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
                if (d>max_A or min_A>d):
                    # 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.. [[Transforms]]!

## 7.1.2 Matrices & Transforms

### Introduction

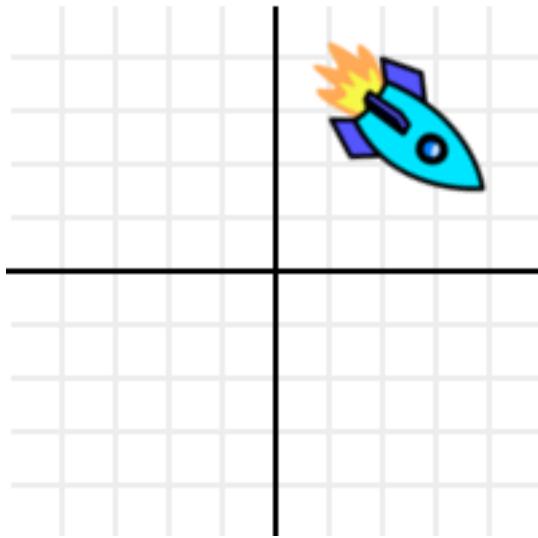
Before reading this tutorial, it is advised to read the previous one about [[Vector Math]] as this one is a direct continuation.

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)

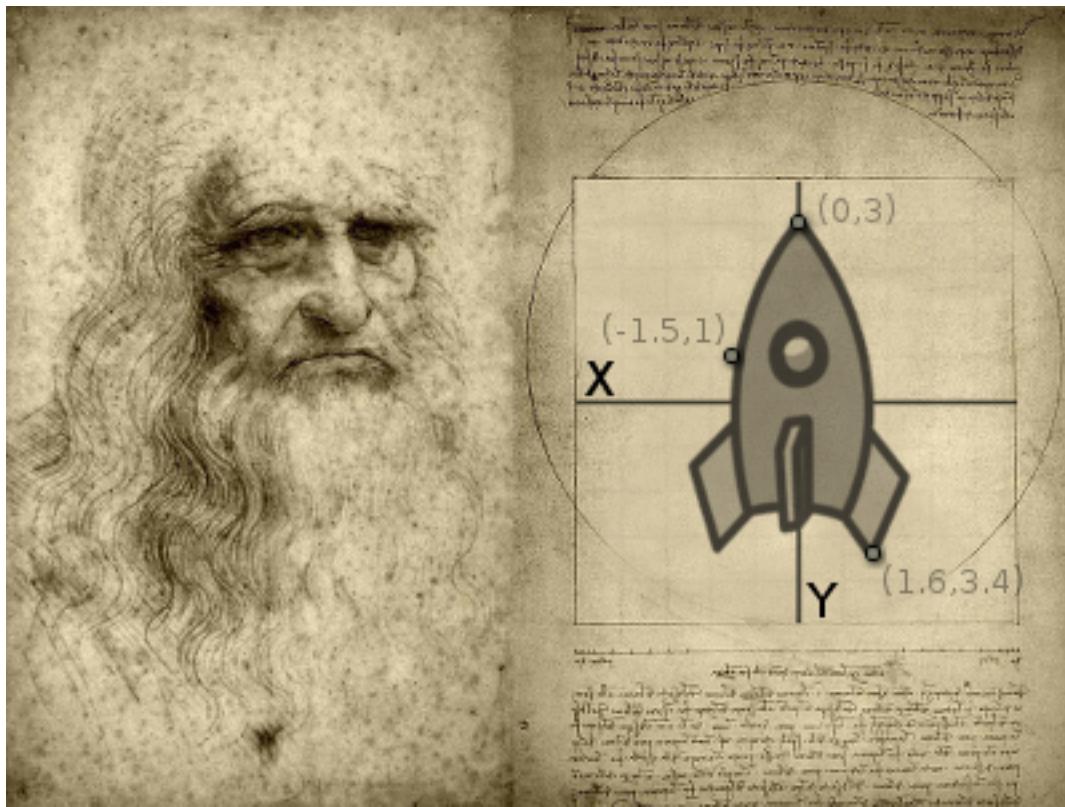
Imagine we have a spaceship somewhere in space. In Godot this is easy, just move the ship somewhere and rotate it:



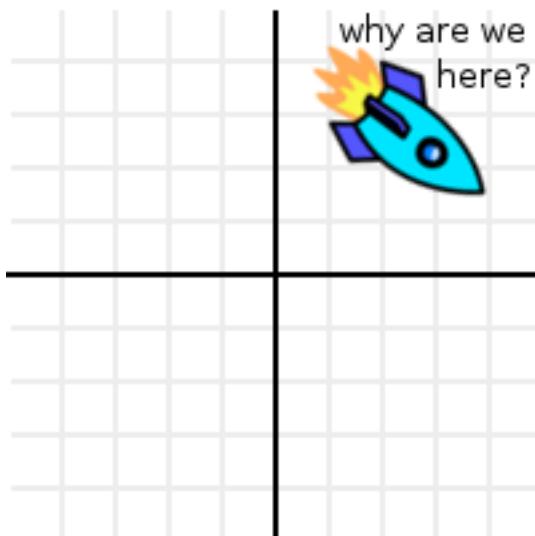
Ok, so in 2D this looks simple, a position and an angle for a rotation. But remember, we are grown ups here and don't use angles (plus, angles are not really even that useful when working in 3D).

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*.



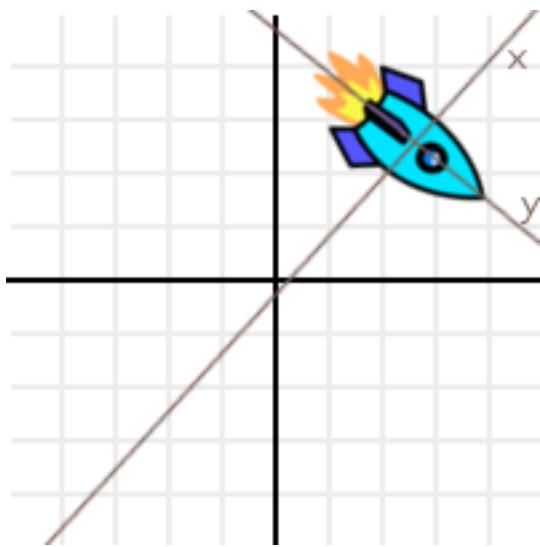
This means that the tip of the ship has a coordinate, the fin has another, etc. Be it in pixels (2D) or vertices (3D). So, let's recall again that the ship was somewhere in space:



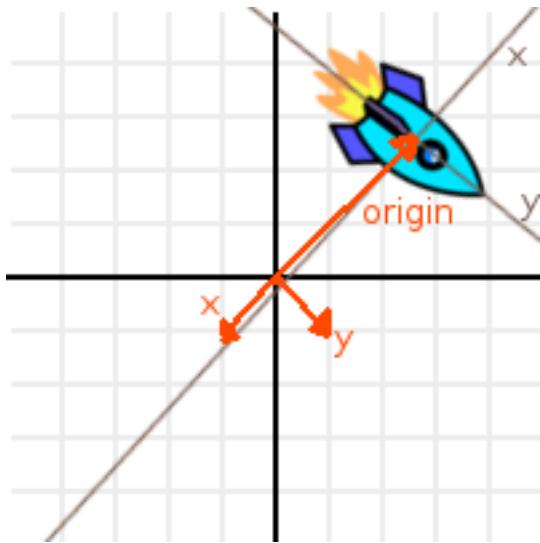
How did it get there? What moved it and rotated it from the place it was designed to its current position? The answer is... a **transform**, the ship was *transformed* from their original position to the new one. This allows the ship to be displayed where it is.

So, a transform is too generic of a term. To solve this puzzle, we will overimpose the ship's original design position at

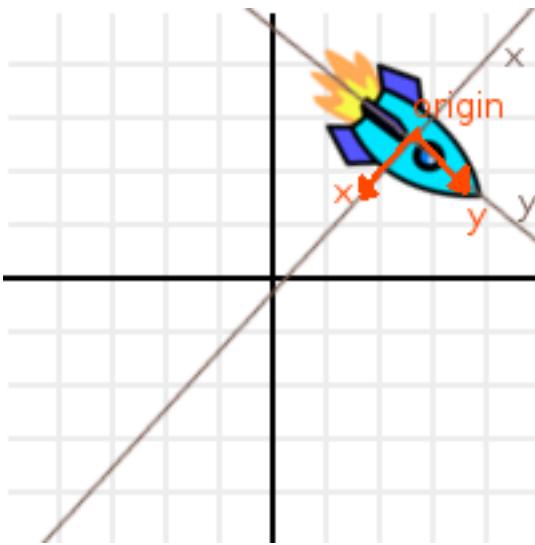
their current position:



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.

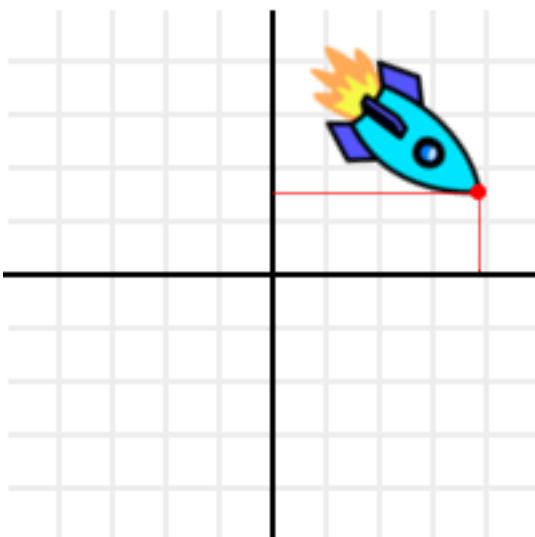


Let’s call the 3 vectors “X”, “Y” and “Origin”, and let’s also overimpose them over the ship so it makes more sense:



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?

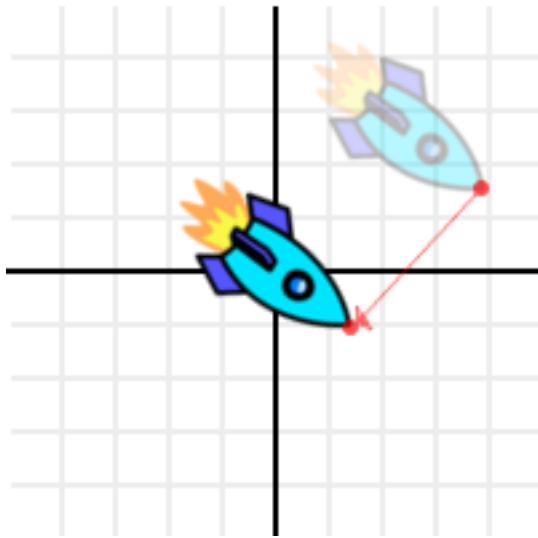
Well, let's take the point from top tip of the ship as reference:



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
```

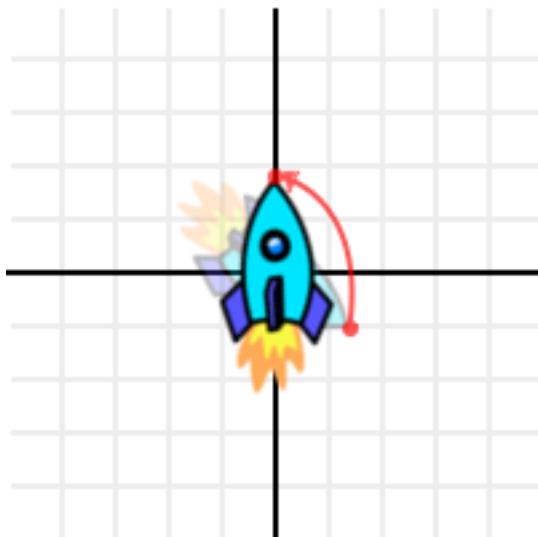
Doing this to the selected point will move it back to the center:



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:

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

Then what we have is.. wait a minute, it's the ship in it's 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 [[tutorial\_vector\_math#distance-to-plane]], 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

The Origin we know what it 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 X and Y of the new, transformed, OCS are pointing towards. It's telling what is in charge of drawing 2D and 3D "Hey, the original X and Y axes or 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*.

$$M = \left\{ \begin{array}{cc} X_x & X_y \\ Y_x & Y_y \end{array} \right\}$$

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 it's 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 it's simplicity.

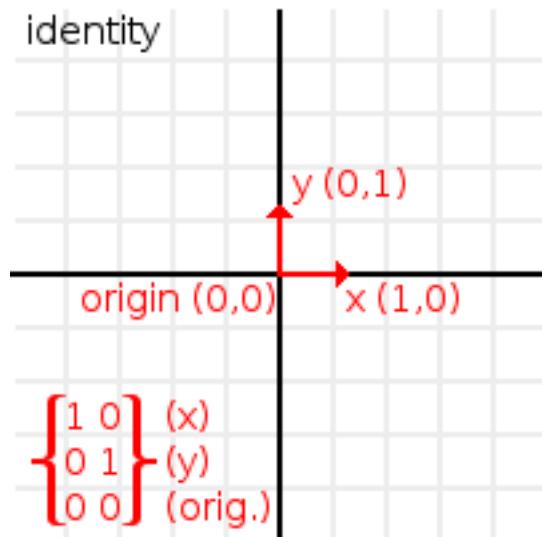
```
var m = Matrix32()
var x = m[0] # 'X'
var y = m[1] # 'Y'
var o = m[2] # 'Origin'
```

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)`



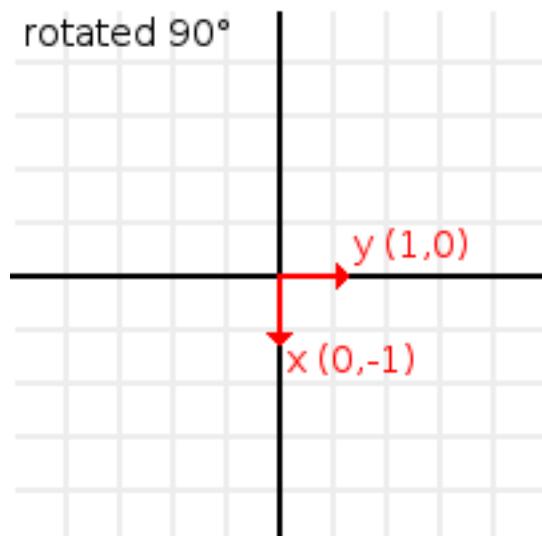
It's easy to guess that an *identity* matrix is just a matrix that aligns the transform to it's parent coordinate system. It's an *OCS* that hasn't been translated, rotated or scaled. All transform types in Godot are created with *identity*.

## Operations

### Rotation

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

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

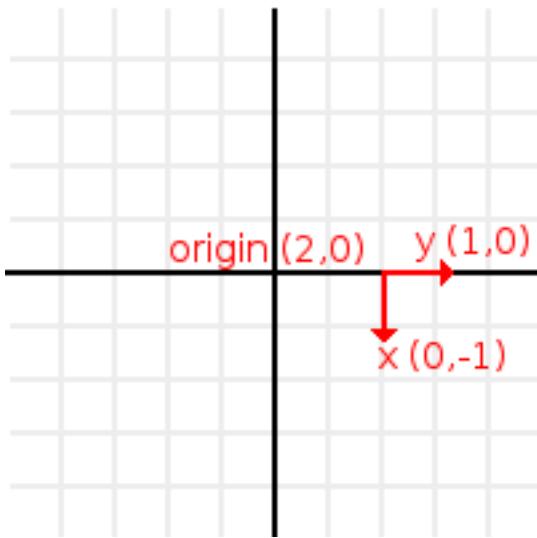


### Translation

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

```
# Move 2 units to the right
var m = Matrix32()
```

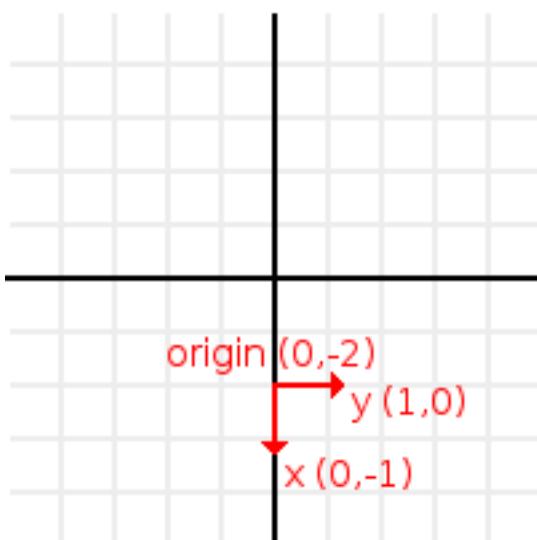
```
m = m.rotated(PI/2) # rotate 90°
m[2] += Vector2(2, 0)
```



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:

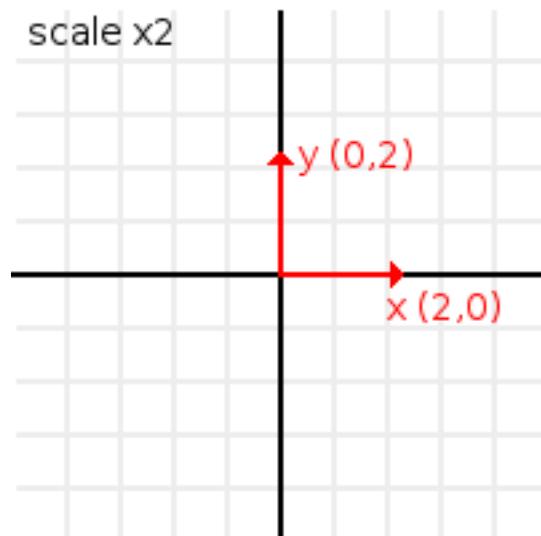
```
# Move 2 units towards where the basis is oriented
var m = Matrix32()
m = m.rotated(PI/2) # rotate 90°
m = m.translated( Vector2(2, 0) )
```



## 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 it's size.  
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 that 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 may 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.

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

And only for basis (no translation):

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

Post - multiplying is also valid:

```
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:

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

Only for Basis:

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

Or pre-multiplication:

```
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 ( $90^\circ$ ), 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:

```
# 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:

```
var mi = m.affine_inverse()
var pos = m.xform(pos)
pos = mi.xform(pos)
#pos is unchanged
```

If the matrix is orthonormal, then:

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

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

To make it a little clearer, this:

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

Is the same as:

```
h1. note the inverse order
pos = (transform2 * transform1).xform(pos)
```

However, this is not the same:

```
# 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

```
# 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:

```
h1. 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:

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

For 3 levels:

```
# 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).

```
# 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:

```
# 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:

```
var m = Matrix3()
var x = m[0] h1. Vector3
var y = m[1] h1. Vector3
var z = m[2] h1. Vector3
```

or, alternatively as:

```
var m = Matrix3()
var x = m.x h1. Vector3
var y = m.y h1. Vector3
var z = m.z h1. Vector3
```

`Matrix3` is also initialized to Identity by default:

$$\left\{ \begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix} \right\} \begin{matrix} (X) \\ (Y) \\ (Z) \end{matrix}$$

## 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).

```
#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:

```
var t = Transform()
pos = t.xform(pos) #transform 3D position
pos = t.basis.xform(pos) h1. (only rotate)
pos = t.origin + pos (only translate)
```

## 7.2 Shaders

### 7.2.1 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. To do so, let's first create a heightmap. To do this, I'll use GIMP editor, 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.

**image0|**

You are now in front of a new, blank, square image.

**image1|**

Then, use a filter to render some clouds on this new image.

**image2|**

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”.

**image3|**

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.

**image4|**

Remove the start cube mesh, then add a new plane to the scene.

**image5|**

Zoom a bit, then switch to Edit mode (Tab key) and in the Tools buttongroup at the left, hit “Subdivide” 5 or 6 times.

**image6|**

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 at the left, click the “Unwrap” button (or simply hit “U”) and select “Smart UV Project”. Keep the default options and hit “Ok”.

**image7|**

Now, we need to switch our view to “UV/Image editor”.

**image8|**

Select all the vertices again (“A”) then in the UV button, select “Export UV Layout”.

**image9|**

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.

**image10|**

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).

**image11**

I like to check “Normals” in the import popup 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://”.

**image12**

Create a MeshInstance node. In the Inspector, load the mesh we just imported. Select “plane.msh” and hit ok.

**image13**

Great! Our plane is now rendered in the 3D view.

**image14**

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.

**image15**

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”.

**image16**

Edit it by clicking the “>” button just right to it. The Shaders editor opens.

**image17**

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 [[Shader]] 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:

```
uniform texture source;
uniform color col;
DIFFUSE = col.rgb * tex(source,UV).rgb;
```

This shader is very simple (it actually comes from the [[Shader]] 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.

**image18**

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,  $0 \times x = 0$ ;). Just change the `col` parameter to another color to see your texture appear:

**image19**

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:

```

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:

**image20**



---

## Asset pipeline

---

### 8.1 General

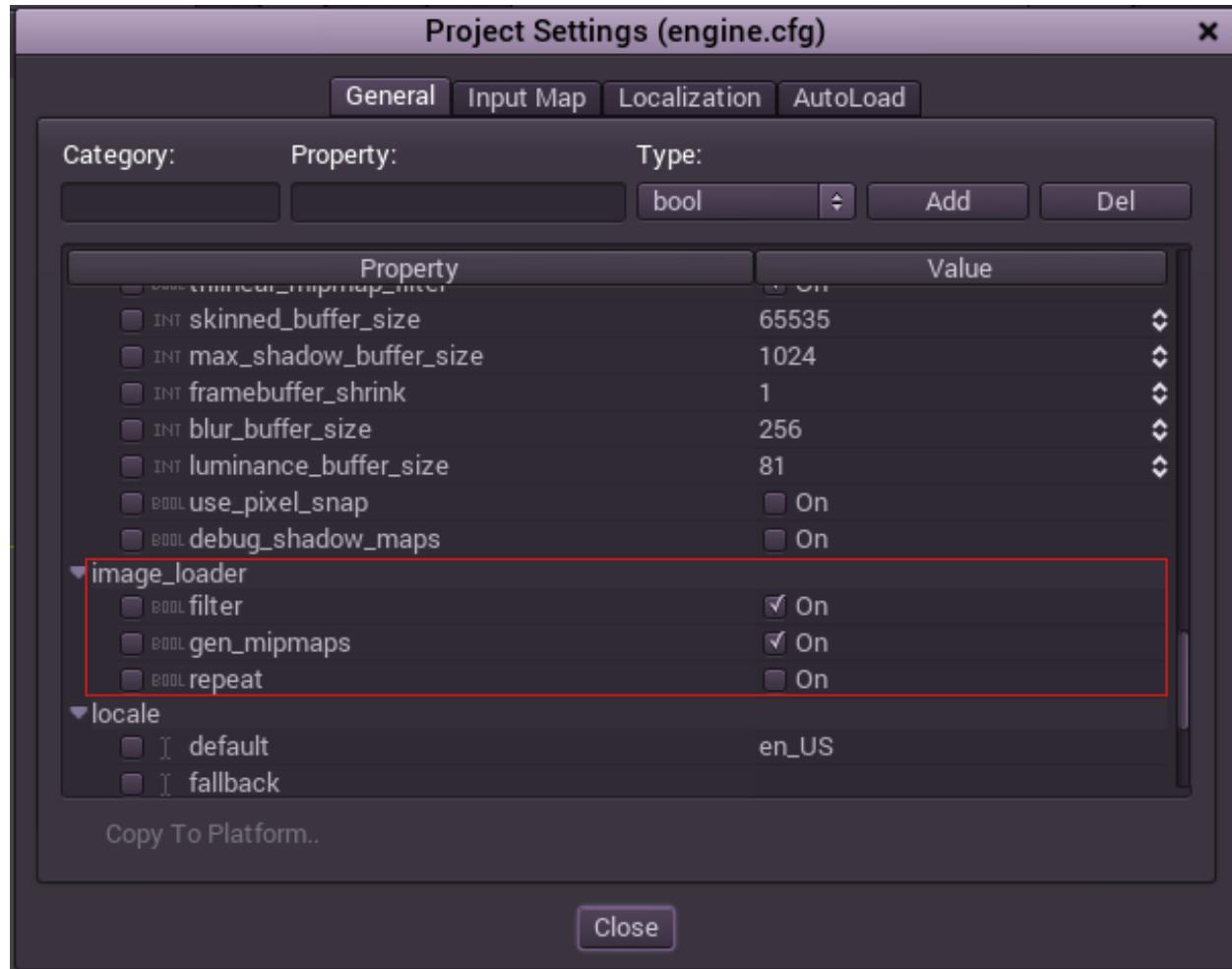
#### 8.1.1 Managing image files

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), mipamps or filtering. Editing this information and saving the texture back will have not 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 correspond to the every image file when loaded:



## Image loader options

### Filter

Filter is used when the image is stretched more than it's 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:



No Filter



Filter

### Repeat

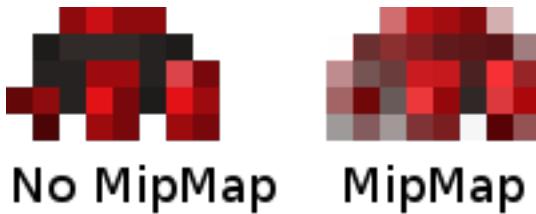
Repeat is mainly used for 3D textures, so it's off by default (textures are imported with the scenes and usually are not in the project as image files). When using UV coordinates (something not as common in 2D), and the UV value goes beyond the 0,0,1,1 rect, the texture repeats instead of clamping to the edge.

## Mipmaps

When the mipmaps option is enabled, Godot will generate mip-maps. 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.



When Mip-Maps are disabled, images start distorting badly when shrunk excessively:



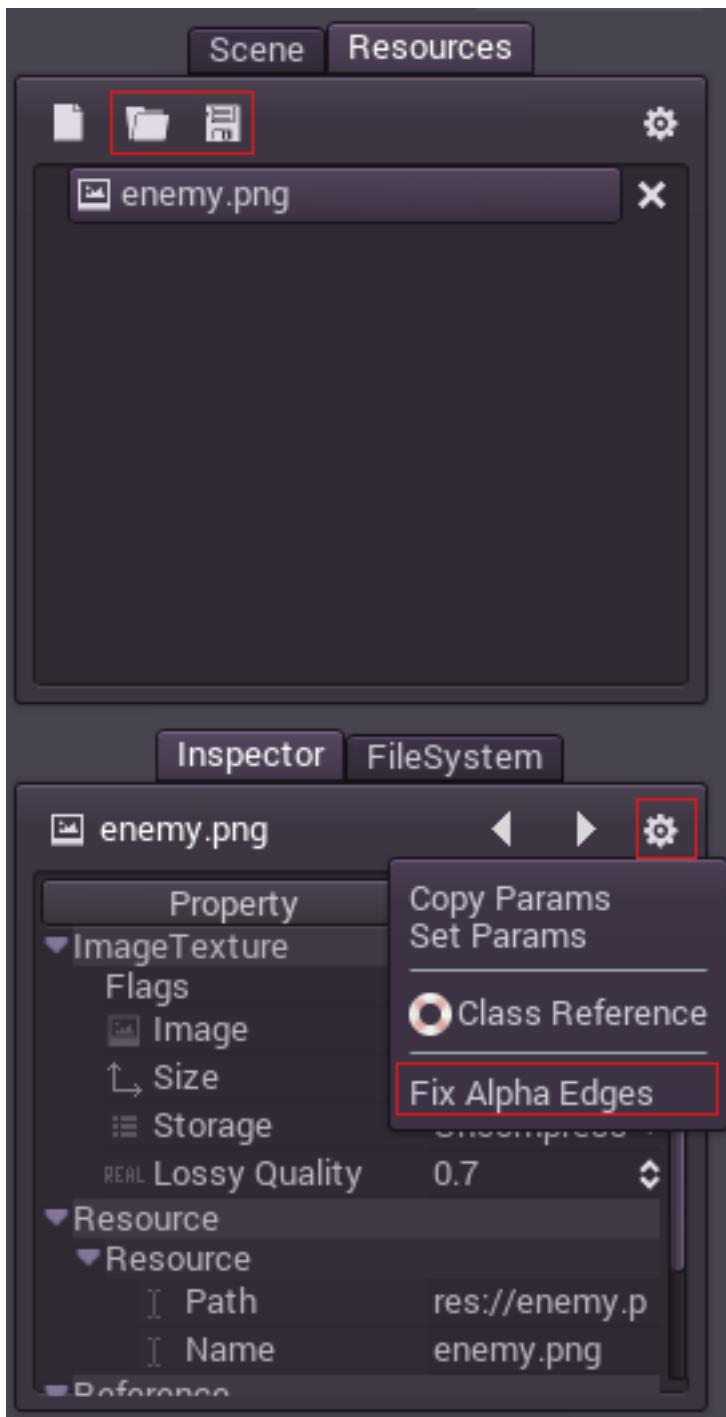
## Alpha blending

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 Border Alpha”, 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 desired 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.

# 8.2 Import

## 8.2.1 Import process

### What is it for?

When Godot was created, it was probably after several failed and not so failed engine attempts (well, each attempt failed a little less.. and so on). One of the most difficult areas of 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 into 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 some 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 for 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 Subversion, Perforce or GIT, is to create the project in a subfolder, so both it and the source assets can be committed to a same repository. For example:

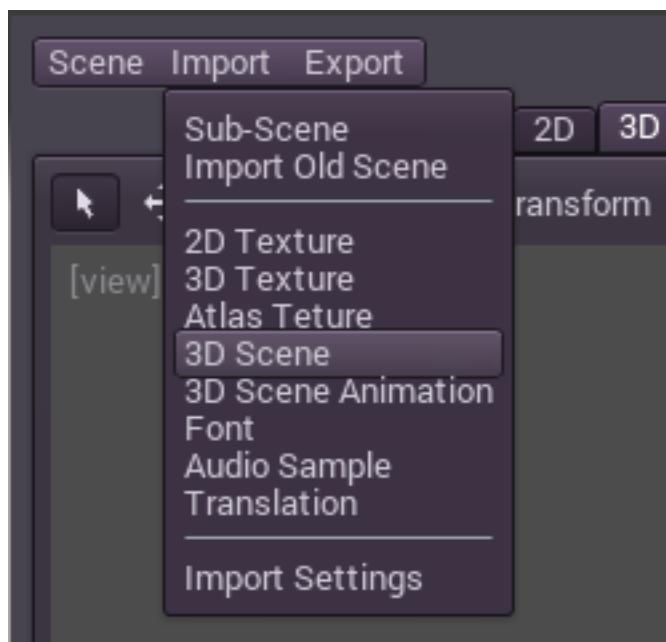
Repository layout:

```
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
```

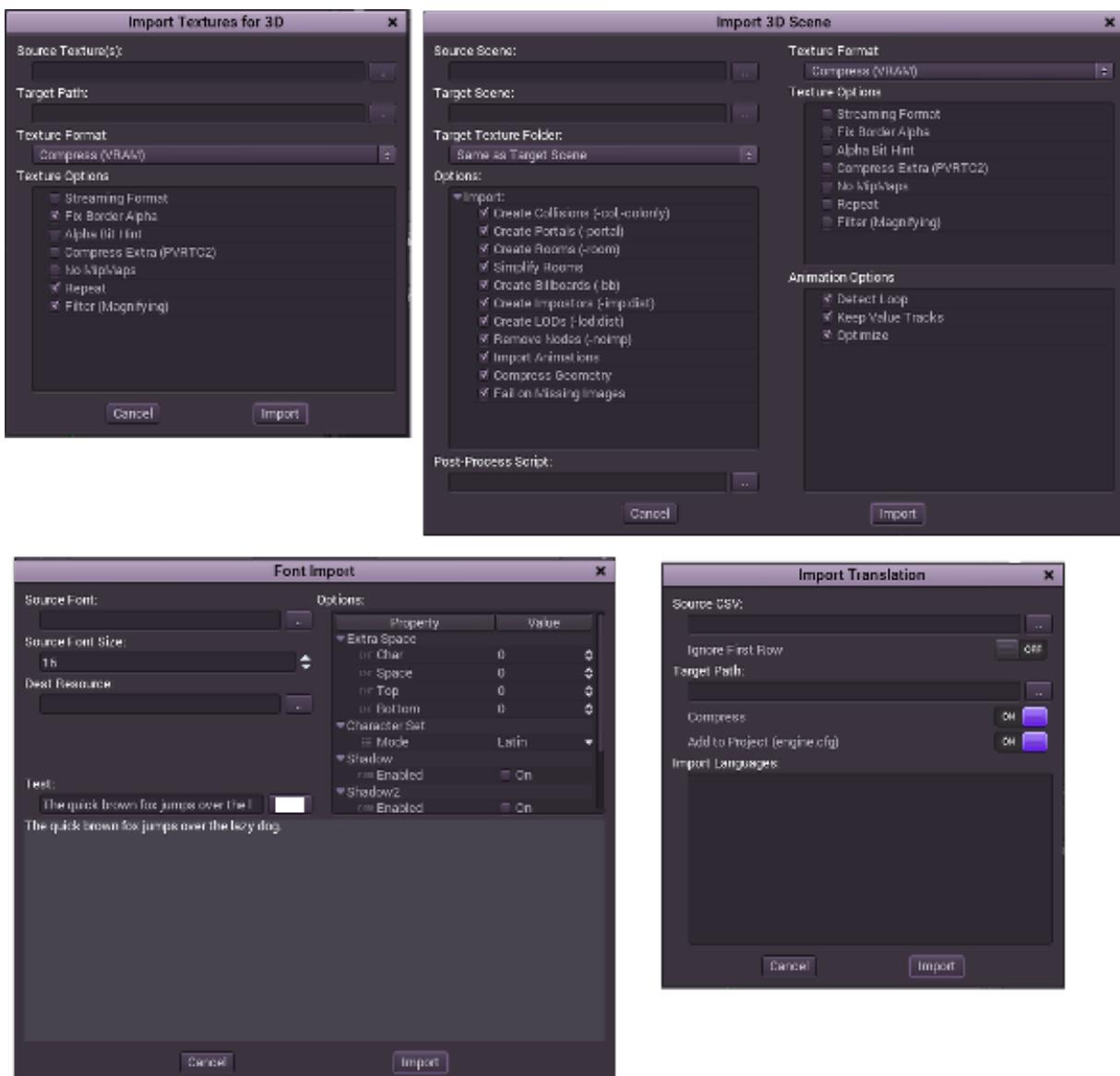
In the above example, artists, musician, translators, etc. can work in the source\_assets/ folder, then import the assets to the game/ folder. When the repository is updated, anyone can re-import the assets if they changed.

#### Import dialogs

Godot provides for importing several types of assets, all of them can be accessed from the import dialog:



Each of the dialog 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](#).

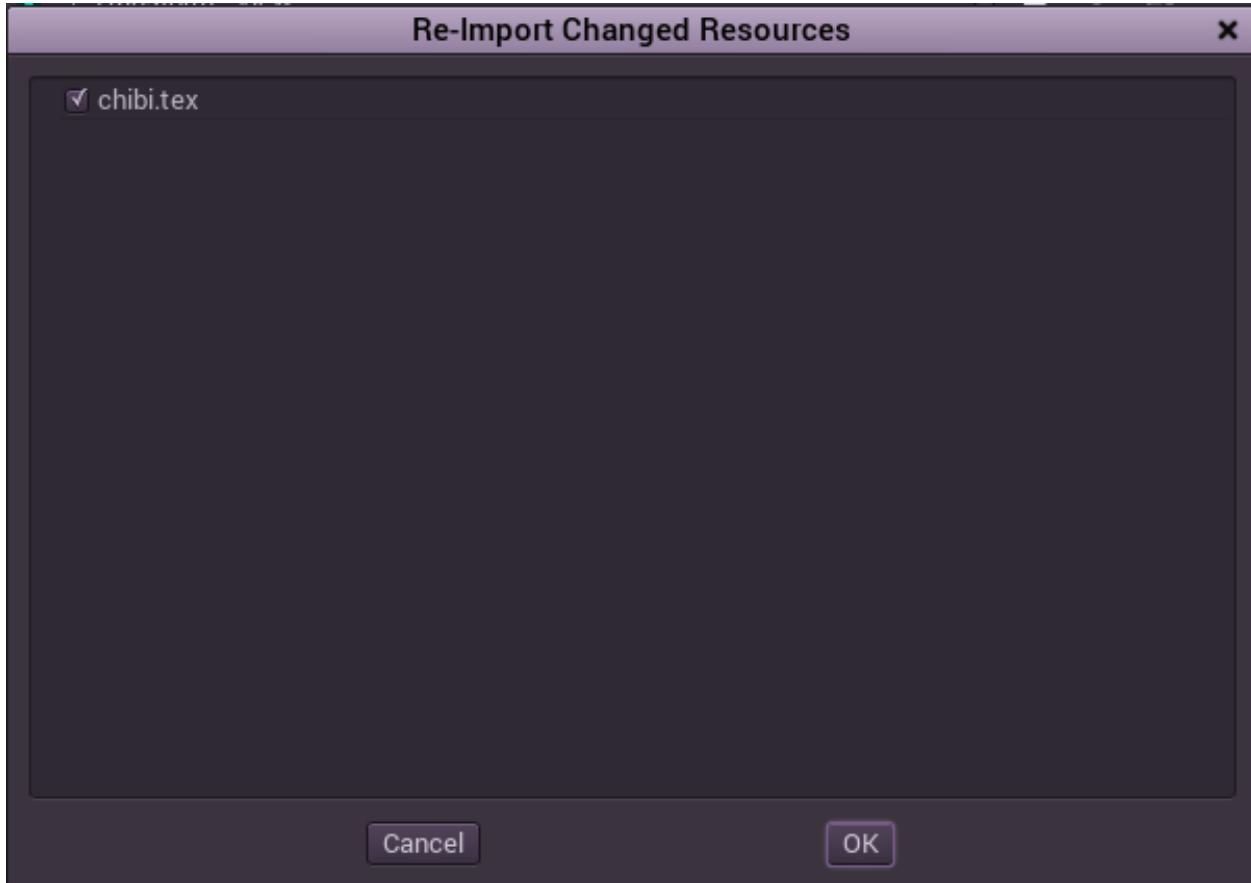
## Tracking changes and re-importing

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.



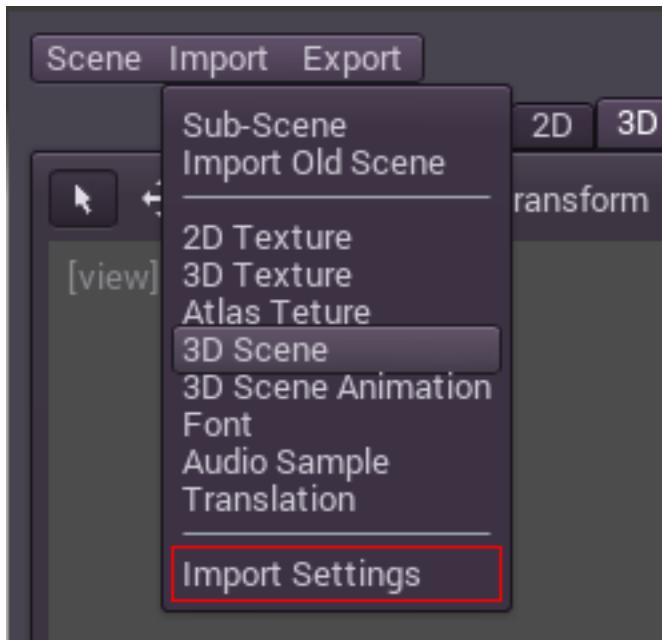
From that moment onward, the user can choose to re-import at any given time by clicking on the red-icon. When this action is done, a dialog will pop-up showing which resources can be re-imported (all selected by default).

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.



## 8.2.2 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 [[Image\_Files|dealing with 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 finer grained control on how textures are imported. If you want to change flags such as repeat, filter, mip-maps, fix edges, etc. **\*PER texture\***, 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.

(Todo, find image sample of why it looks bad)

To solve this, mipsmaps 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 (ie 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):



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

Asides 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.

Video texture compression formats are several and non 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 2D games and UIs because it has visible visual artifacts. This is specially noticeable on games that use the trendy victory social game artwork. However, again, 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 mipmapmaps for optimum stretching? What if we have a lot of images in different random sizes? All will have to be scaled and mipmapped when loaded (using more CPU and memory) or when imported (using more memory). 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).

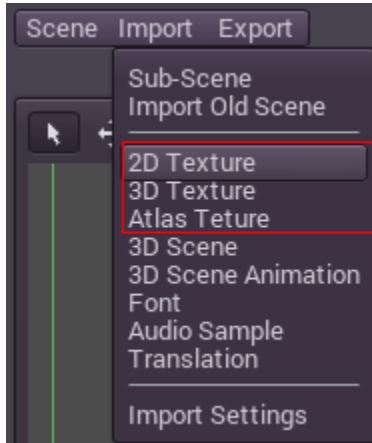
As a small plus, atlases can decrease the amount of “state changes” when drawing. If a lot of objects that are drawn using several different textures are converted to atlas, then the texture rebinds per object will go from dozens or hundreds to one. This will give the performance a small boost.

## Artists use PSD

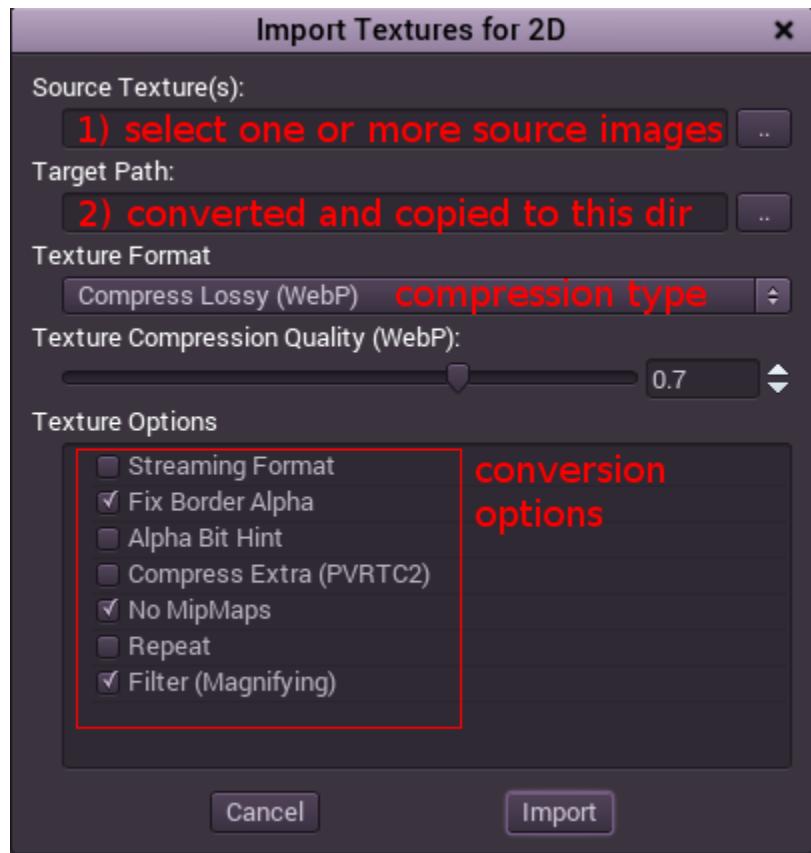
Still wondering whether to use the texture importer or not? Remember that in the end, artists will often use Photoshop anyway, so it may be wiser to just let the import subsystem to take care of importing and converting the PSD files instead of asking the artist to save a png and copy it to the project 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)

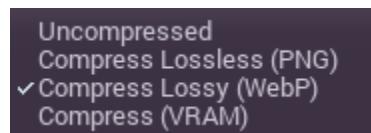
One or more source images can be selected from the same folder (this importer can do batch-conversion). This can be from inside or outside the project.

#### 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):



Each of the four options described in this table together with their advantages and disadvantages ( **image5** = Best, **image6** = Worst ):

	Uncompressed	Compress Lossless (PNG)	Compress Lossy (WebP)	Compress VRAM
Description	Stored as raw pixels	Stored as PNG	Stored as WebP	Stored as S3TC/BC,PVRTC/ETC, depending on platform
Size on Disk	 Large	 Small	 Very Small	 Small
Memory Usage	 Large	 Large	 Large	 Small
Performance	 Normal	 Normal	 Normal	 Fast
Quality Loss	 None	 None	 Slight	 Moderate
Load Time	 Normal	 Slow	 Slow	 Fast

### 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).

### 8.2.3 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

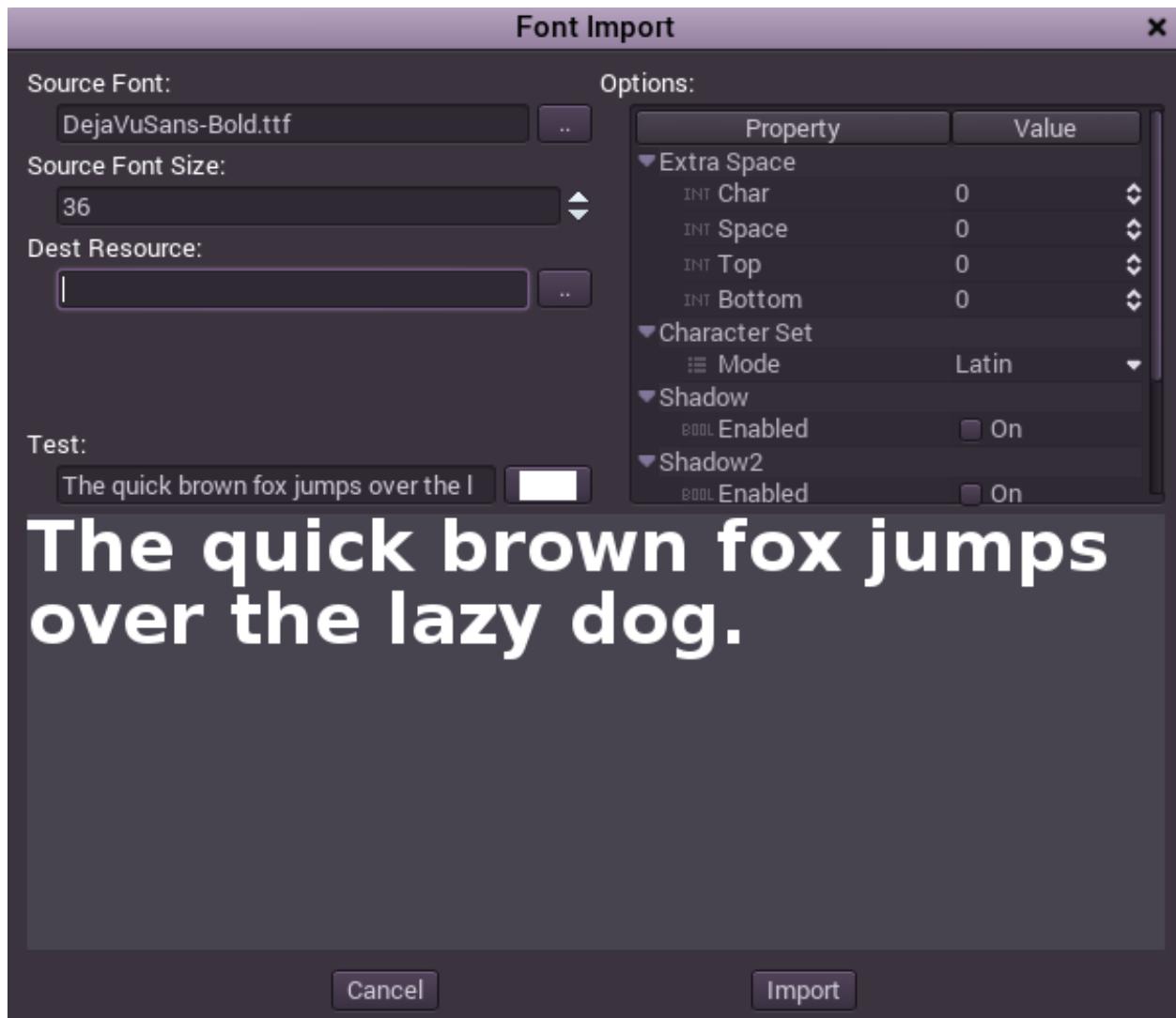
This conversion process consists of rendering a vector font to a given point size and storing all the resulting characters in a bitmap texture. The bitmap texture is then used by the GPU to draw a small quad for each character and form readable strings.



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 :). The options which 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.

The quick brown fox jumps over the lazy dog.  
The quick brown fox jumps over the lazy dog.  
The quick brown fox jumps over the lazy dog.

### Shadows & outline

Fonts can be added a shadow. For this, the font is drawn again below on a different color and 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.

**The quick brown fox jumps over the lazy dog.**  
**The quick brown fox jumps over the lazy dog.**

### 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) contains 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 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.

## 8.2.4 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 copying 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.

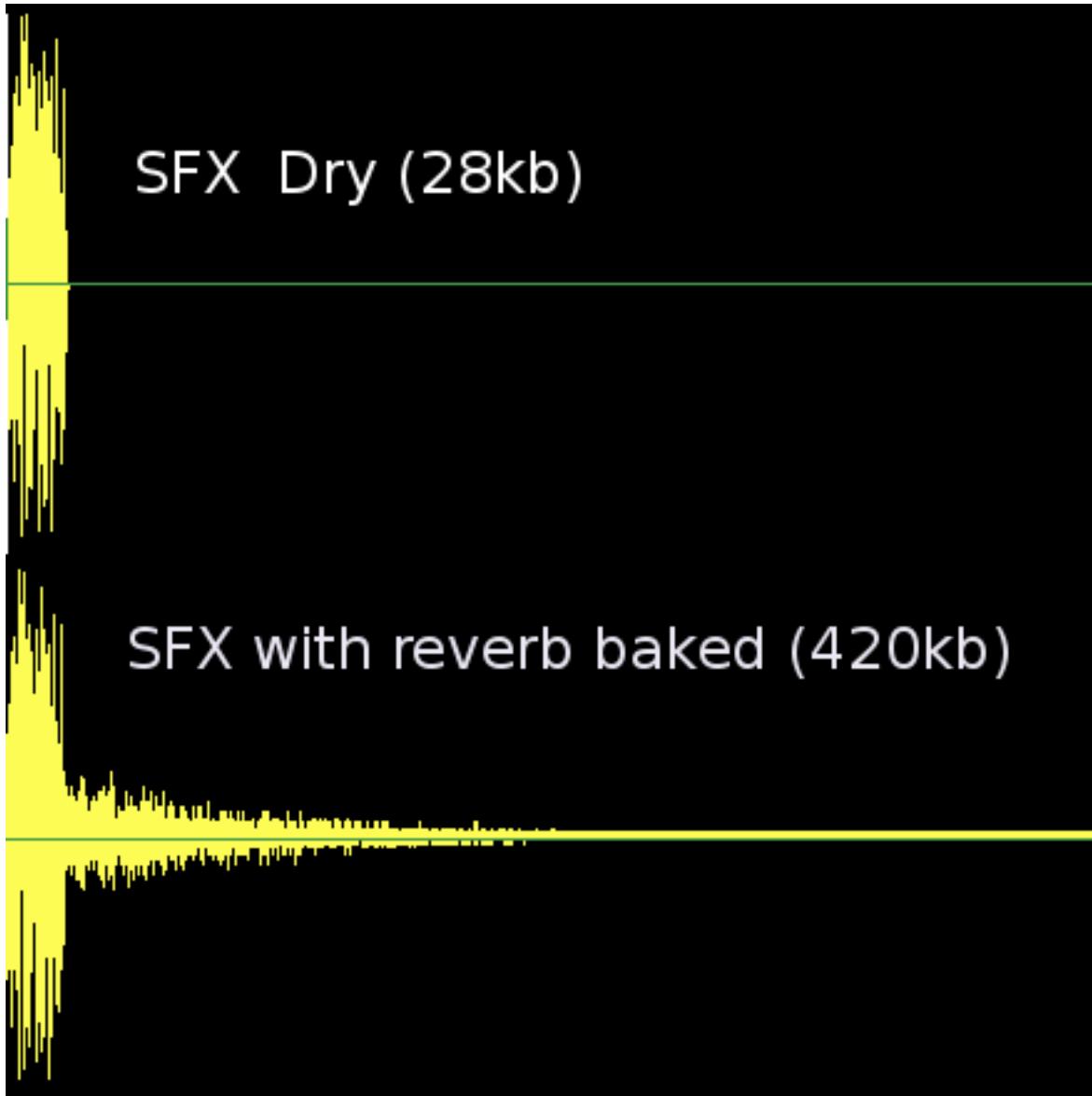
As is usual with graphics, the situation where programmers don’t really know about audio and audio engineers don’t know about programming is also common in the industry. This leads to a scenario where a project ends up wasting resources unnecessarily.

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 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 take a long time to fade out. Finally, many DAWs also add silence at the beginning when normalizing to wav.

This results in extremely large files to integrate more often than desired, 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!



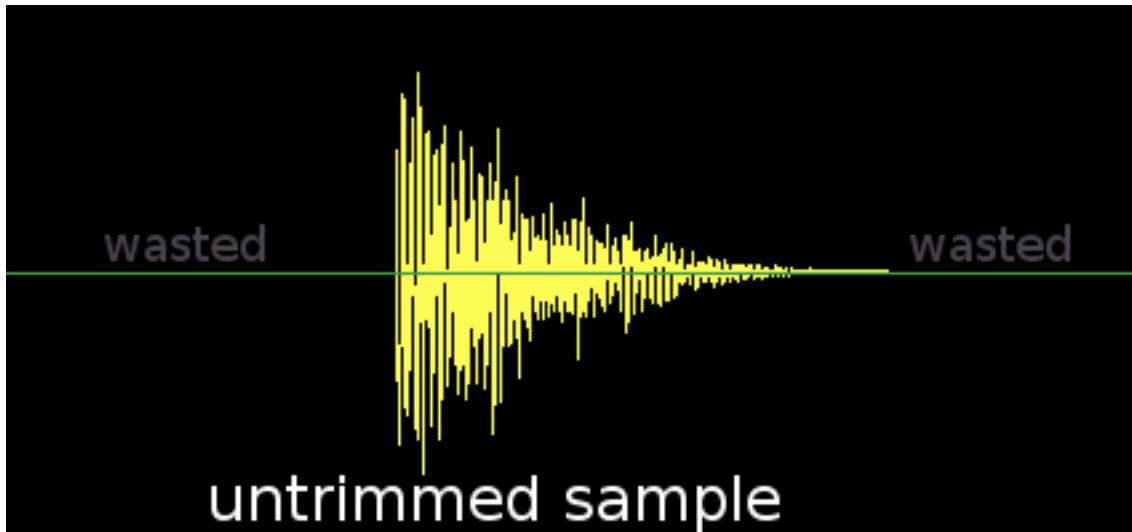
Another common problem is that, while it's useful for working inside a DAW, high dynamic range (24 bits) and high sampling rate (96khz) is completely unnecessary for use in a game, as there is no [audible difference](#). If positional sound is going to be used (for 2D and 3D), the panning and stereo 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:

Format	1 Second of Audio	Frame Size
24 bits, 96 khz, Stereo	576kb	12
16 bits, 44 khz, Mono	88kb	2
16 bits, IMA-ADPCM	22kb	1/2

As seen, for being no audible difference, the 16 bits, 44khz takes *6 times less memory* than the 24 bits, 96khz, Stereo version. The IMA-ADPCM version 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. As a plus, several samples can be loaded and batch-converted, just like textures.

### 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.

## 8.2.5 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 multi-language text.

There are two approaches to generate multi language games and applications. Both are based on a key:value system. The first is to use one of the languages as 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 to revise the text while it's being translated to others. the unique ID can be a number, a string, or a string with a number (it's just a unique string anyway).

Translators also, most of the time prefer to work with spreadsheets (either as a Microsoft Excel file or a shared Google Spreadsheet).

## Translation format

To complete the picture and allow efficient support for translations, Godot has a special importer that can read .csv files. Both Microsoft Excel and Google Spreadsheet can export to this format, so the only requirement is that the files have a special format. The csv files must be saved in utf-8 encoding and the format is as follows:

KEY1	string	string	string
KEY2	string	string	string
KEYN	string	string	string

The “lang” tags must represent a language, it 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 esta?	
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 that 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 to load and remove translations at runtime, too.

## 8.3 Export

### 8.3.1 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 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 use 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. In other words, 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, which is the fact that 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)

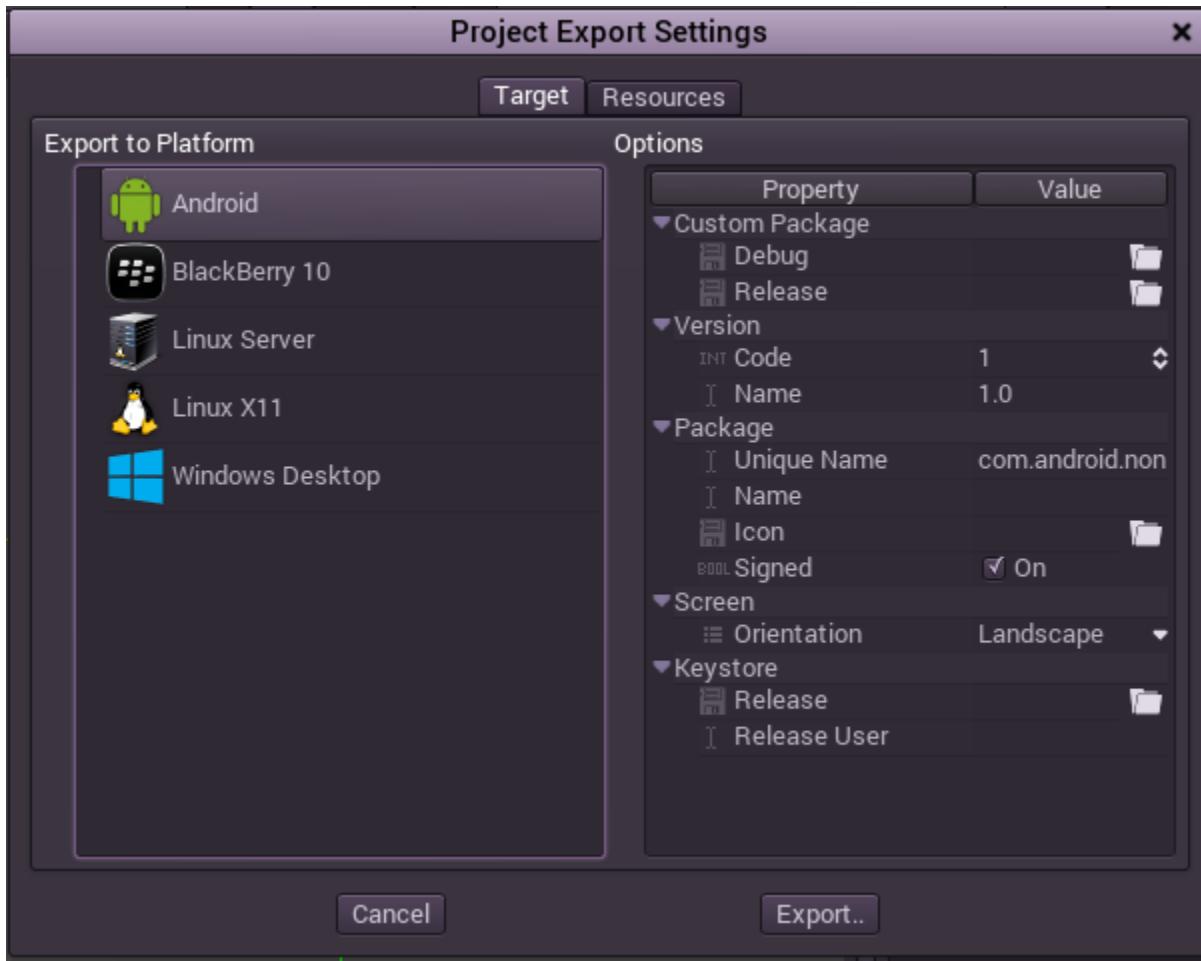
### Export Dialog

After many attempts at different export workflows, the current one has worked the best. At the time of this writing, not all platforms are supported yet, but that will change soon.

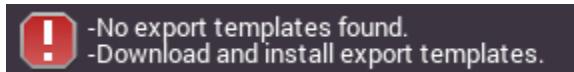
To open the export dialog, just click the “Export” Button:



The dialog will open, showing all the supported export platforms:



The default options are often enough to export, so tweaking them is not necessary until it's needed. However, many platforms require additional tools (SDKs) to be installed to be able to export. Additionally, Godot needs exports 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 he or she resolves it:

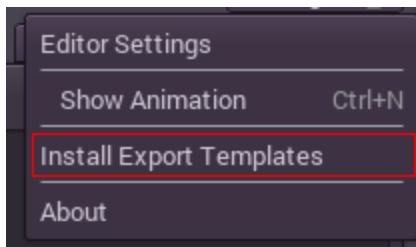


At that time, the user is expected to come back to the wiki and follow instructions on how to properly set up that platform.

### Export Templates

Apart from setting up the platform, the export templates must be installed to be able to export projects. They can be downloaded as a .tpz (a renamed .zip) file from the wiki.

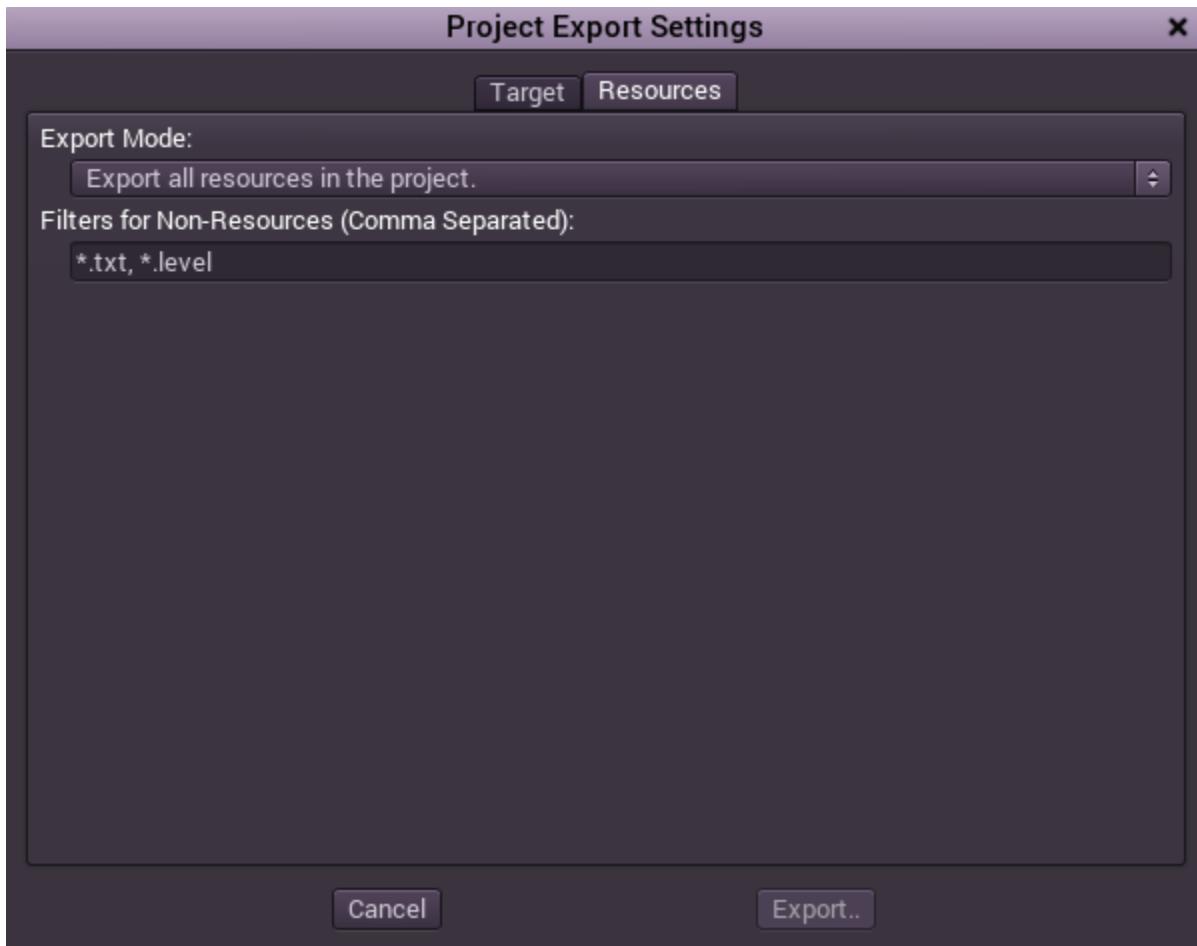
Once downloaded, they can be installed using the “Install Export Templates” option in the editor:



### 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:

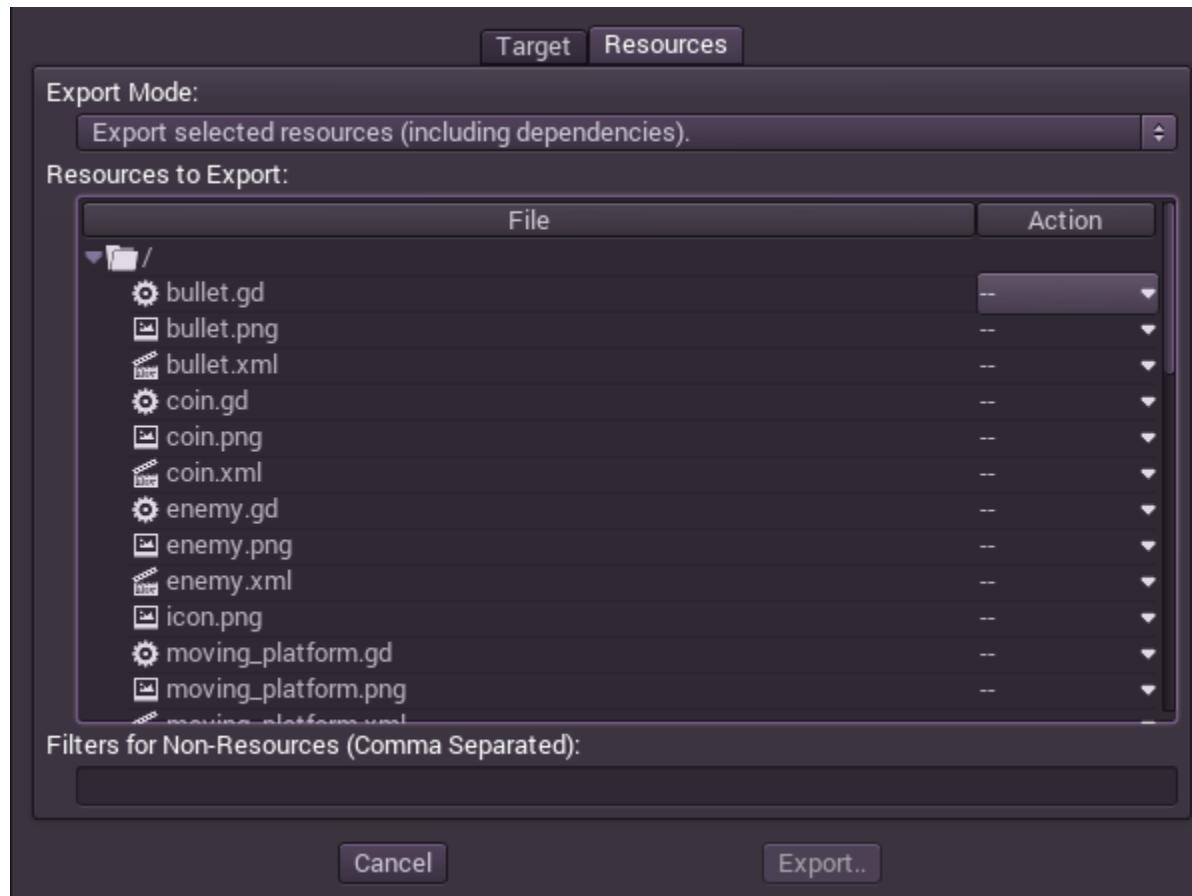
- Export every single file in the project
- Export only resources (+custom filter), this is default.
- Export only selected resources (+custom filter)



- **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 dir, 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.



### 8.3.2 One-click deploy

#### Sounds Good, What is it?

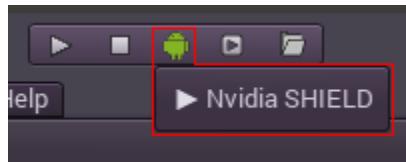
This feature will pop up automatically once a platform is properly configured and a supported device is connected to the computer. Since things can go wrong at many levels (platform may not be configured correctly, SDK may incorrectly installed, device may be improperly configured, kitty ate the USB 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, like the computer, usb is recognized, usb cable is plugged, etc).
3. Connect the device..
4. And Voila!



Click once.. and deploy!

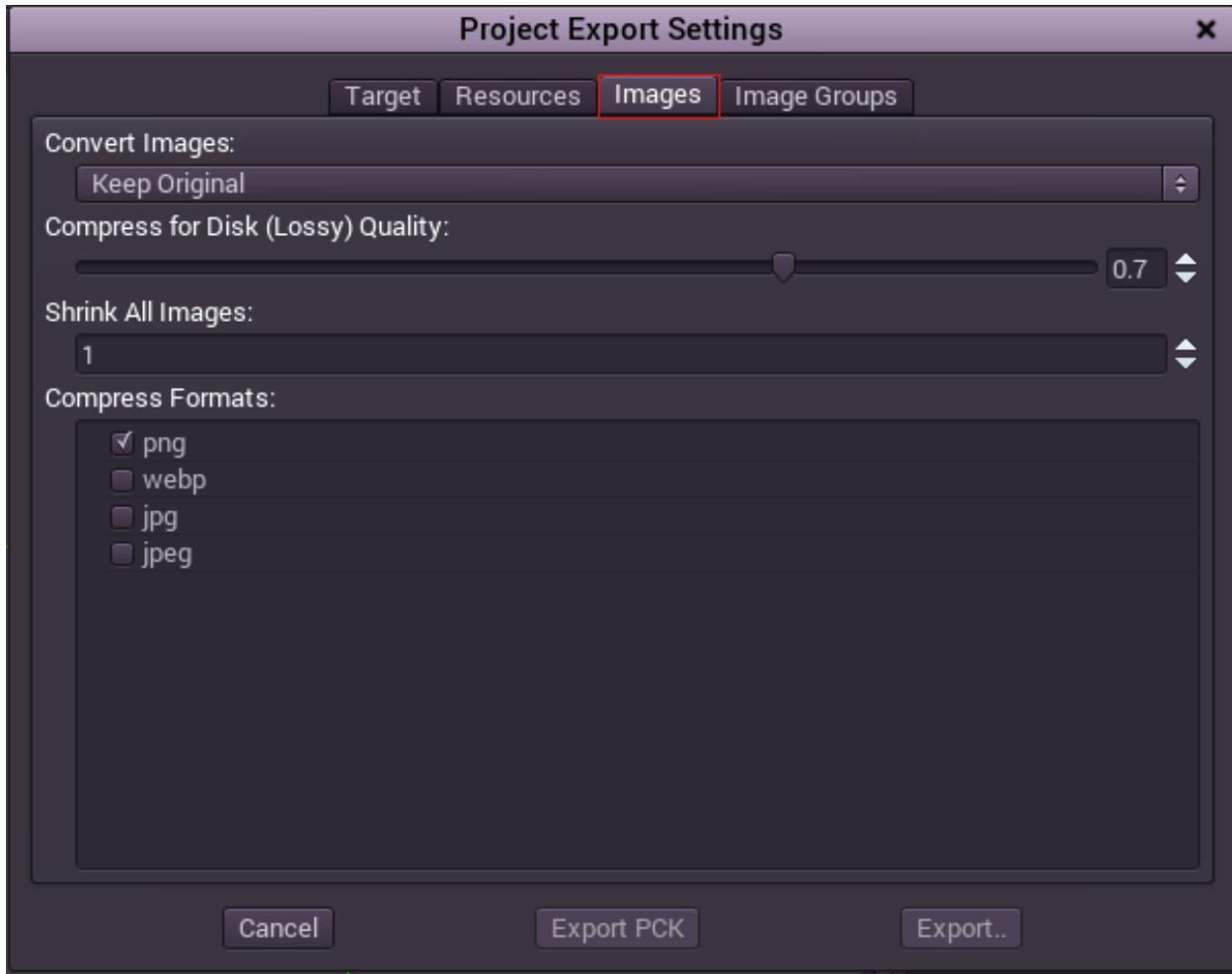
### 8.3.3 Exporting images

It is often desired to do an operation to 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 -> web) 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 Export Dialog, go to the Images tab:



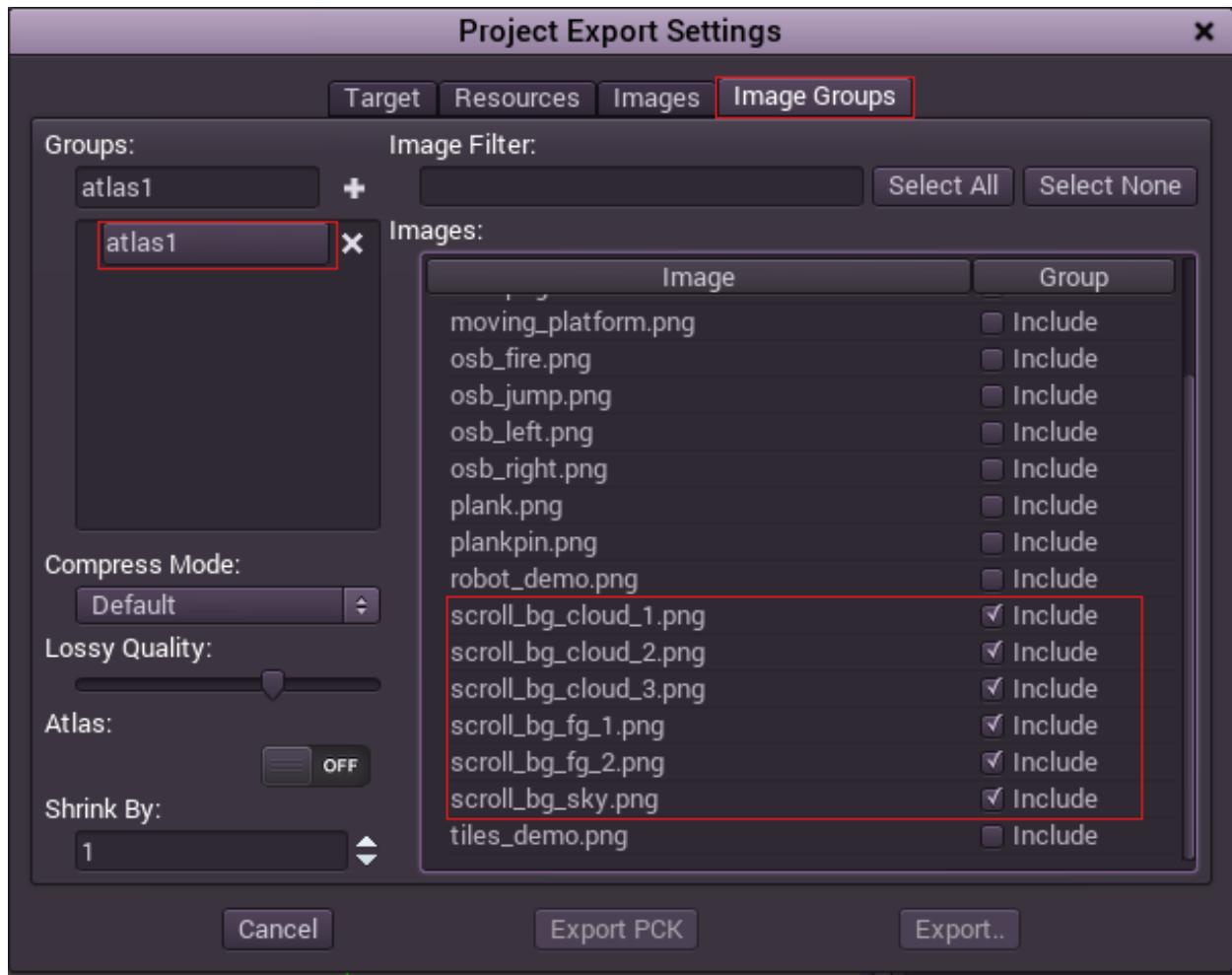
In this dialog the image extensions for conversion can be selected, and operations can be performed that apply to all images (except those in groups -next section for that-):

- **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 an 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 the same time. So if the image is in another group different to the current one being edited, it will not be selectable.



## Atlas

As a plus, an atlas can be created from a group. 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 (through 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.

### 8.3.4 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 work properly.

### 8.3.5 Exporting for Android

Exporting for android has much less requirements than compiling Godot for it. As follows are the steps 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>

#### Download the Java 6 or OpenJDK6

Download and install Java 6 or OpenJDK 6, Android needs this version and it seems that jarsigner (what is used to sign APKs) from greater versions do not work.

#### 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:

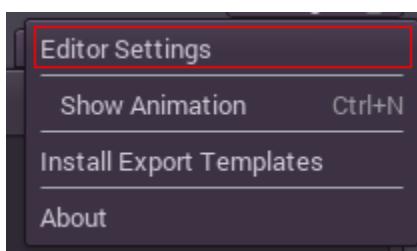
```
keytool -keyalg RSA -genkeypair -alias androiddebugkey -keypass android -keystore debug.keystore -storepass android -dname "CN=Android Debug,O=Android,C=US" -validity 9999
```

#### Make sure you have adb

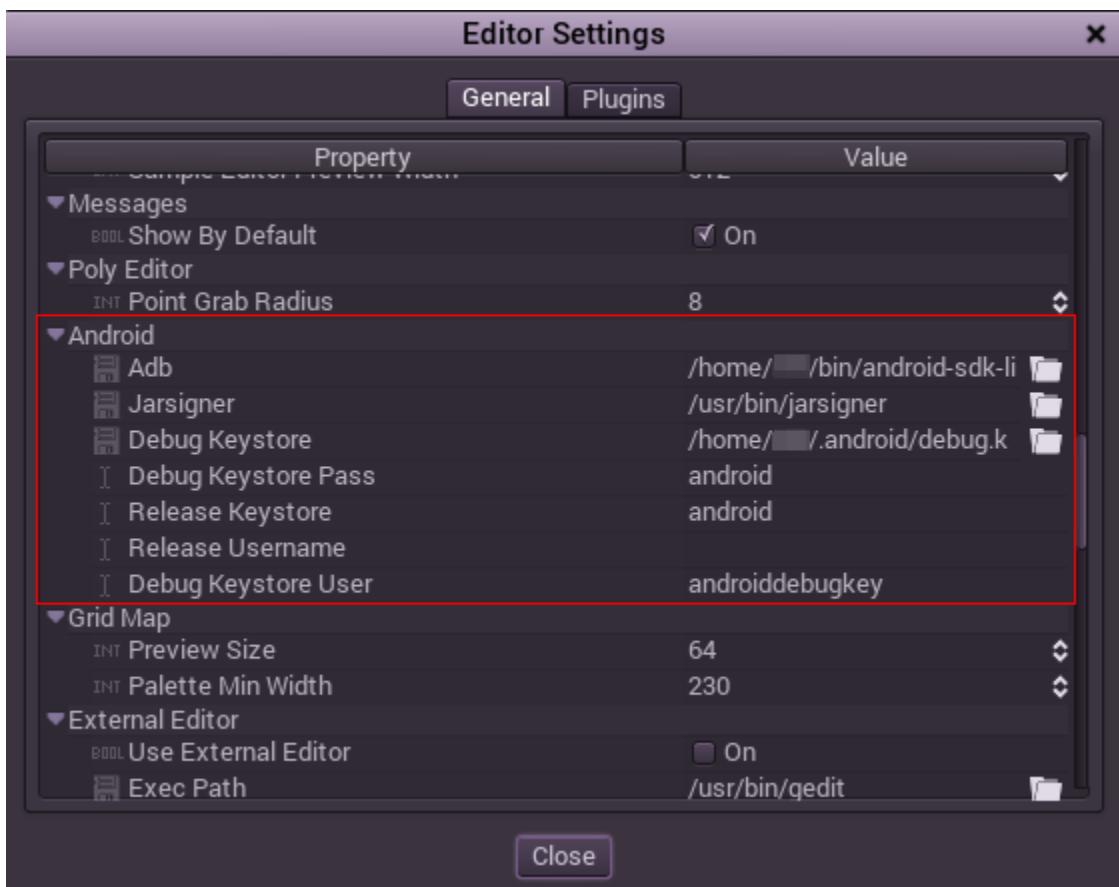
ADB is the command line tool used to communicate with Android devices. It's installed with the SDK, but you may need to install one (any) of the Android API levels for it to be installed in the SDK directory.

#### Setting it up in Godot

Enter the Editor Settings screen. This screens 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 JDK6)
- The debug *keystore*

Once that is configured, everything is ready to export to Android!

### 8.3.6 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

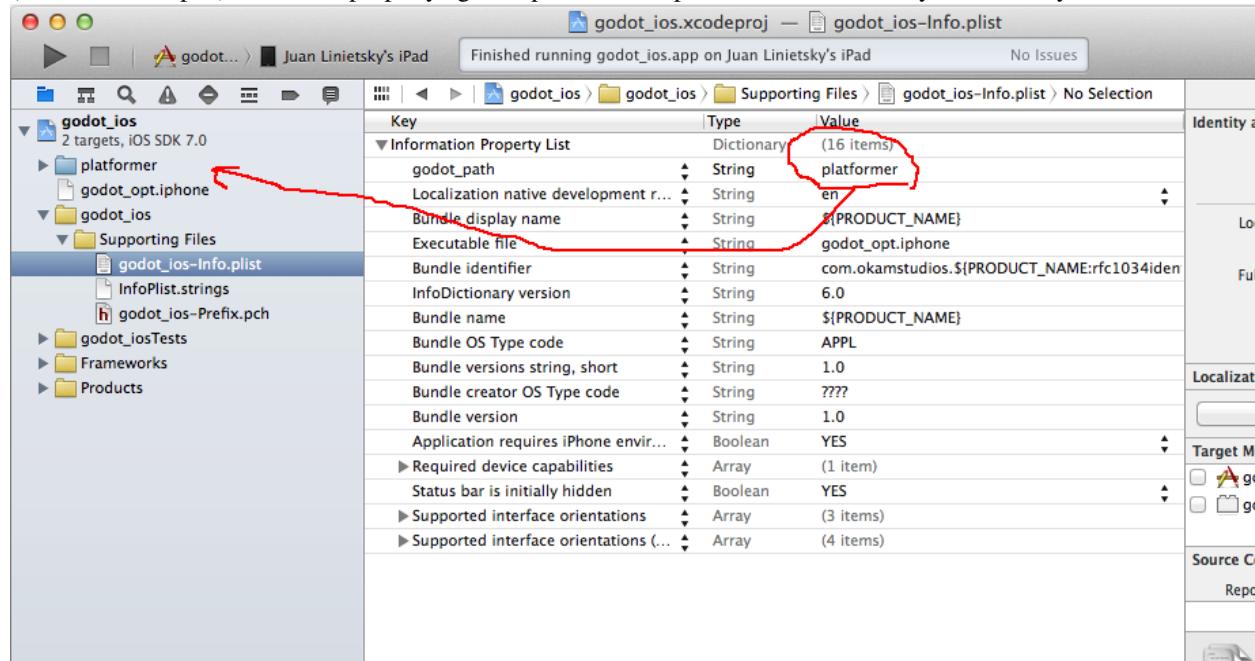
- Download XCode for iOS
- Download the export templates: <http://www.godotengine.org/projects/godot-engine/documents>
- Since there is no automatic deployer yet, unzip export\_templates.tpz manually and extract GodotiOSXCode.zip from it.

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\_pclexport your project for Windows]], 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.



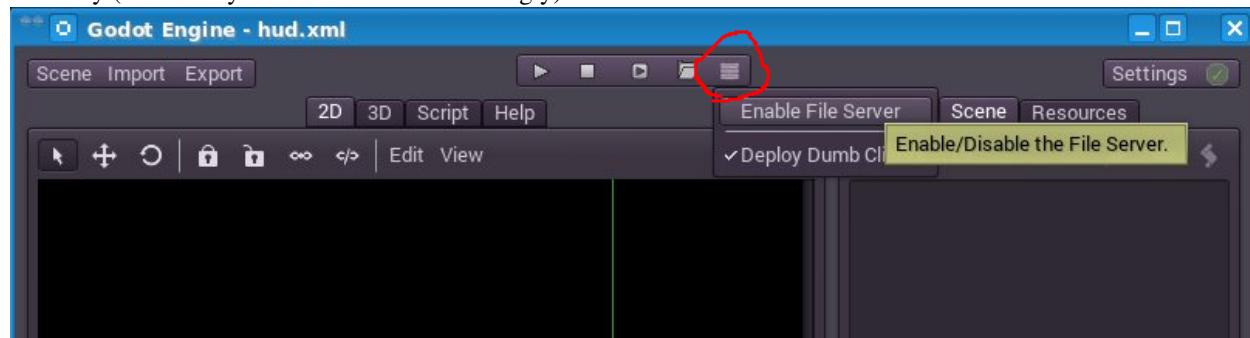
Alternatively you can add all the files from your game directly, with “engine.cfg” at the root.

## 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.

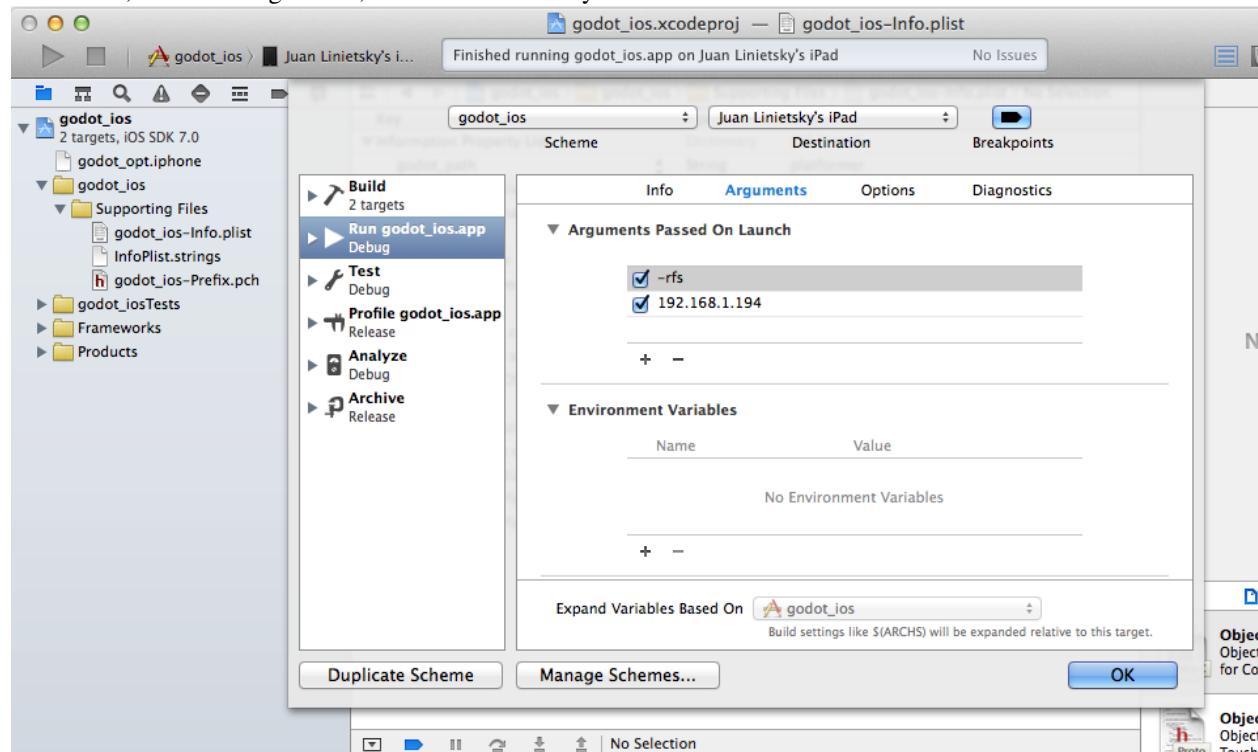
### Setting up the file host

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.



---

**Classes reference**

---

## 9.1 @GDScript

**Category:** Core

### 9.1.1 Brief Description

Built-in GDScript functions.

### 9.1.2 Member Functions

<i>float</i>	<i>sin (float s )</i>
<i>float</i>	<i>cos (float s )</i>
<i>float</i>	<i>tan (float s )</i>
<i>float</i>	<i>sinh (float s )</i>
<i>float</i>	<i>cosh (float s )</i>
<i>float</i>	<i>tanh (float s )</i>
<i>float</i>	<i>asin (float s )</i>
<i>float</i>	<i>acos (float s )</i>
<i>float</i>	<i>atan (float s )</i>
<i>float</i>	<i>atan2 (float x, float y )</i>
<i>float</i>	<i>sqrt (float s )</i>
<i>float</i>	<i>fmod (float x, float y )</i>
<i>float</i>	<i>fposmod (float x, float y )</i>
<i>float</i>	<i>floor (float s )</i>
<i>float</i>	<i>ceil (float s )</i>
<i>float</i>	<i>round (float s )</i>
<i>float</i>	<i>abs (float s )</i>
<i>float</i>	<i>sign (float s )</i>
<i>float</i>	<i>pow (float x, float y )</i>
<i>float</i>	<i>log (float s )</i>
<hr/>	
<i>float</i>	<i>exp (float s )</i>
<i>float</i>	<i>isnan (float s )</i>
<i>float</i>	<i>isinf (float s )</i>
<i>float</i>	<i>ease (float s, float curve )</i>

Continued on next page

Table 9.1 – continued from previous page

<i>float</i>	<i>decimals</i> ( <i>float</i> step )
<i>float</i>	<i>stepify</i> ( <i>float</i> s, <i>float</i> step )
<i>float</i>	<i>lerp</i> ( <i>float</i> from, <i>float</i> to, <i>float</i> weight )
<i>float</i>	<i>dectime</i> ( <i>float</i> value, <i>float</i> amount, <i>float</i> step )
<i>Nil</i>	<i>randomize</i> ()
<i>int</i>	<i>randi</i> ()
<i>float</i>	<i>randf</i> ()
<i>float</i>	<i>rand_range</i> ( <i>float</i> from, <i>float</i> to )
<i>Nil</i>	<i>seed</i> ( <i>float</i> seed )
<i>Array</i>	<i>rand_seed</i> ( <i>float</i> seed )
<i>float</i>	<i>deg2rad</i> ( <i>float</i> deg )
<i>float</i>	<i>rad2deg</i> ( <i>float</i> rad )
<i>float</i>	<i>linear2db</i> ( <i>float</i> nrg )
<i>float</i>	<i>db2linear</i> ( <i>float</i> db )
<i>float</i>	<i>max</i> ( <i>float</i> a, <i>float</i> b )
<i>float</i>	<i>min</i> ( <i>float</i> a, <i>float</i> b )
<i>float</i>	<i>clamp</i> ( <i>float</i> val, <i>float</i> min, <i>float</i> max )
<i>int</i>	<i>nearest_po2</i> ( <i>int</i> val )
<i>WeakRef</i>	<i>weakref</i> ( <i>Object</i> obj )
<i>FuncRef</i>	<i>funcref</i> ( <i>Object</i> instance, <i>String</i> funcname )
<i>Object</i>	<i>convert</i> ( Variant what, <i>int</i> type )
<i>int</i>	<i>typeof</i> ( Variant what )
<i>String</i>	<i>str</i> ( Variant what, Variant ... )
<i>Nil</i>	<i>print</i> ( Variant what, Variant ... )
<i>Nil</i>	<i>printt</i> ( Variant what, Variant ... )
<i>Nil</i>	<i>prints</i> ( Variant what, Variant ... )
<i>Nil</i>	<i>printerr</i> ( Variant what, Variant ... )
<i>Nil</i>	<i>printraw</i> ( Variant what, Variant ... )
<i>String</i>	<i>var2str</i> ( Variant var )
<i>Variant</i>	<i>str2var</i> ( <i>String</i> string )
<i>RawArray</i>	<i>var2bytes</i> ( Variant var )
<i>Variant</i>	<i>bytes2var</i> ( <i>RawArray</i> bytes )
<i>Array</i>	<i>range</i> ( Variant ... )
<i>Resource</i>	<i>load</i> ( <i>String</i> path )
<i>Dictionary</i>	<i>inst2dict</i> ( <i>Object</i> inst )
<i>Object</i>	<i>dict2inst</i> ( <i>Dictionary</i> dict )
<i>int</i>	<i>hash</i> ( Variant var:Variant )
<i>Color</i>	<i>Color8</i> ( <i>int</i> r8, <i>int</i> g8, <i>int</i> b8, <i>int</i> a8 )
<i>Nil</i>	<i>print_stack</i> ()
<i>Object</i>	<i>instance_from_id</i> ( <i>int</i> instance_id )

### 9.1.3 Numeric Constants

- **PI = 3.141593** — Constant that represents how many times the diameter of a circumference fits around its perimeter.

### 9.1.4 Description

This contains the list of built-in gdscript functions. Mostly math functions and other utilities. Everything else is expanded by objects.

### 9.1.5 Member Function Description

- `float sin (float s)`

Standard sine function.

- `float cos (float s)`

Standard cosine function.

- `float tan (float s)`

Standard tangent function.

- `float sinh (float s)`

Hyperbolic sine.

- `float cosh (float s)`

Hyperbolic cosine.

- `float tanh (float s)`

Hyperbolic tangent.

- `float asin (float s)`

Arc-sine.

- `float acos (float s)`

Arc-cosine.

- `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.

- `float sqrt (float s)`

Square root.

- `float fmod (float x, float y)`

Module (remainder of x/y).

- `float fposmod (float x, float y)`

Module (remainder of x/y) that wraps equally in positive and negative.

- `float floor (float s)`

Floor (rounds down to nearest integer).

- `float ceil (float s)`

Ceiling (rounds up to nearest integer).

- `float round (float s)`

Round to nearest integer.

- `float abs (float s)`

Remove sign (works for integer and float).

- `float sign (float s)`

Return sign (-1 or +1).

- `float pow ( float x, float y )`

Power function, x elevate to y.

- `float log ( float s )`

Natural logarithm.

- `float exp ( float s )`

Exponential logarithm.

- `float isnan ( float s )`

Return true if the float is not a number.

- `float isinf ( float s )`

Return true if the float is infinite.

- `float 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.

- `float decimals ( float step )`

Return the amount of decimals in the floating point value.

- `float stepify ( float s, float step )`

Snap float value to a given step.

- `float lerp ( float from, float to, float weight )`

Linear interpolates between two values by a normalized value.

- `float dectime ( float value, float amount, float step )`

Decreases time by a specified amount.

- `Nil randomize ()`

Reset the seed of the random number generator with a new, different one.

- `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.`

- `float randf ()`

Random value (0 to 1 float).

- `float rand_range ( float from, float to )`

Random range, any floating point value between ‘from’ and ‘to’

- `Nil seed ( float seed )`
- `Array rand_seed ( float seed )`

Random from seed, pass a seed and an array with both number and new seed is returned.

- `float deg2rad ( float deg )`

Convert from degrees to radians.

- `float rad2deg ( float rad )`

Convert from radians to degrees.

- `float linear2db ( float nrg )`

Convert from linear energy to decibels (audio).

- `float db2linear ( float db )`

Convert from decibels to linear energy (audio).

- `float max ( float a, float b )`

Return the maximum of two values.

- `float min ( float a, float b )`

Return the minimum of two values.

- `float clamp ( float val, float min, float max )`

Clamp both values to a range.

- `int nearest_po2 ( int val )`

Return the nearest larger power of 2 for an integer.

- `WeakRef weakref ( Object obj )`

Return a weak reference to an object.

- `FuncRef funcref ( Object instance, String funcname )`

Returns a reference to the specified function

- `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](#).

- `int typeof ( Variant what )`

Returns the internal type of the given Variant object, using the TYPE\_\* enum in [@Global Scope](#).

- `String str ( Variant what, Variant ... )`

Convert one or more arguments to strings in the best way possible.

- `Nil print ( Variant what, Variant ... )`

Print one or more arguments to strings in the best way possible to a console line.

- `Nil printt ( Variant what, Variant ... )`

Print one or more arguments to the console with a tab between each argument.

- `Nil prints ( Variant what, Variant ... )`

- `Nil printer ( Variant what, Variant ... )`

Print one or more arguments to strings in the best way possible to standard error line.

- `Nil prinraw ( Variant what, Variant ... )`

Print one or more arguments to strings in the best way possible to console. No newline is added at the end.

- `String var2str ( Variant var )`

Converts the value of a variable to a String.

- `Variant str2var ( String string )`

- *RawArray* **var2bytes** ( Variant var )
- Variant **bytes2var** ( *RawArray* bytes )
- *Array* **range** ( Variant ... )

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).

- *Resource* **load** ( *String* path )

Load a resource from the filesystem, pass a valid path as argument.

- *Dictionary* **inst2dict** ( *Object* inst )

Convert a script class instance to a dictionary (useful for serializing).

- *Object* **dict2inst** ( *Dictionary* dict )

Convert a previously converted instances to dictionary back into an instance. Useful for deserializing.

- *int* **hash** ( Variant var:Variant )

Hashes the variable passed and returns an integer.

- *Color* **Color8** ( *int* r8, *int* g8, *int* b8, *int* a8 )
- *Nil* **print\_stack** ( )

Print a stack track at code location, only works when running with debugger turned on.

- *Object* **instance\_from\_id** ( *int* instance\_id )

## 9.2 @Global Scope

**Category:** Core

### 9.2.1 Brief Description

Global scope constants and variables.

### 9.2.2 Member Variables

- *Performance* **Performance**
- *Globals* **Globals**
- *IP* **IP**
- *Geometry* **Geometry**
- *ResourceLoader* **ResourceLoader**
- *ResourceSaver* **ResourceSaver**
- *PathRemap* **PathRemap**
- *OS* **OS**
- *Reference* **Marshalls**
- *TranslationServer* **TranslationServer**

- *TranslationServer* **TS**
- *Input* **Input**
- *InputMap* **InputMap**
- *VisualServer* **VisualServer**
- *VisualServer* **VS**
- *AudioServer* **AudioServer**
- *AudioServer* **AS**
- *PhysicsServer* **PhysicsServer**
- *PhysicsServer* **PS**
- *Physics2DServer* **Physics2DServer**
- *Physics2DServer* **PS2D**
- *SpatialSoundServer* **SpatialSoundServer**
- *SpatialSoundServer* **SS**
- *SpatialSound2DServer* **SpatialSound2DServer**
- *SpatialSound2DServer* **SS2D**

### 9.2.3 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**
- **KEY\_RETURN = 16777221**
- **KEY\_ENTER = 16777222**

- **KEY\_INSERT = 16777223**
- **KEY\_DELETE = 16777224**
- **KEY\_PAUSE = 16777225**
- **KEY\_PRINT = 16777226**
- **KEY\_SYSREQ = 16777227**
- **KEY\_CLEAR = 16777228**
- **KEY\_HOME = 16777229**
- **KEY\_END = 16777230**
- **KEY\_LEFT = 16777231**
- **KEY\_UP = 16777232**
- **KEY\_RIGHT = 16777233**
- **KEY\_DOWN = 16777234**
- **KEY\_PAGEUP = 16777235**
- **KEY\_PAGEDOWN = 16777236**
- **KEY\_SHIFT = 16777237**
- **KEY\_CONTROL = 16777238**
- **KEY\_META = 16777239**
- **KEY\_ALT = 16777240**
- **KEY\_CAPSLOCK = 16777241**
- **KEY\_NUMLOCK = 16777242**
- **KEY\_SCROLLLOCK = 16777243**
- **KEY\_F1 = 16777244**
- **KEY\_F2 = 16777245**
- **KEY\_F3 = 16777246**
- **KEY\_F4 = 16777247**
- **KEY\_F5 = 16777248**
- **KEY\_F6 = 16777249**
- **KEY\_F7 = 16777250**
- **KEY\_F8 = 16777251**
- **KEY\_F9 = 16777252**
- **KEY\_F10 = 16777253**
- **KEY\_F11 = 16777254**
- **KEY\_F12 = 16777255**
- **KEY\_F13 = 16777256**
- **KEY\_F14 = 16777257**
- **KEY\_F15 = 16777258**

- **KEY\_F16 = 16777259**
- **KEY\_KP\_ENTER = 16777344**
- **KEY\_KP\_MULTIPLY = 16777345**
- **KEY\_KP\_DIVIDE = 16777346**
- **KEY\_KP\_SUBSTRACT = 16777347**
- **KEY\_KP\_PERIOD = 16777348**
- **KEY\_KP\_ADD = 16777349**
- **KEY\_KP\_0 = 16777350**
- **KEY\_KP\_1 = 16777351**
- **KEY\_KP\_2 = 16777352**
- **KEY\_KP\_3 = 16777353**
- **KEY\_KP\_4 = 16777354**
- **KEY\_KP\_5 = 16777355**
- **KEY\_KP\_6 = 16777356**
- **KEY\_KP\_7 = 16777357**
- **KEY\_KP\_8 = 16777358**
- **KEY\_KP\_9 = 16777359**
- **KEY\_SUPER\_L = 16777260**
- **KEY\_SUPER\_R = 16777261**
- **KEY\_MENU = 16777262**
- **KEY\_HYPER\_L = 16777263**
- **KEY\_HYPER\_R = 16777264**
- **KEY\_HELP = 16777265**
- **KEY\_DIRECTION\_L = 16777266**
- **KEY\_DIRECTION\_R = 16777267**
- **KEY\_BACK = 16777280**
- **KEY\_FORWARD = 16777281**
- **KEY\_STOP = 16777282**
- **KEY\_REFRESH = 16777283**
- **KEY\_VOLUMEDOWN = 16777284**
- **KEY\_VOLUMEMUTE = 16777285**
- **KEY\_VOLUMEUP = 16777286**
- **KEY\_BASSBOOST = 16777287**
- **KEY\_BASSUP = 16777288**
- **KEY\_BASSDOWN = 16777289**
- **KEY\_TREBLEUP = 16777290**

- **KEY\_TREBLEDOWN** = 16777291
- **KEY\_MEDIAPLAY** = 16777292
- **KEY\_MEDIASTOP** = 16777293
- **KEY\_MEDIAPREVIOUS** = 16777294
- **KEY\_MEDIANEXT** = 16777295
- **KEY\_MEDIARECORD** = 16777296
- **KEY\_HOMEPAGE** = 16777297
- **KEY\_FAVORITES** = 16777298
- **KEY\_SEARCH** = 16777299
- **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
- **KEY\_EXCLAM** = 33
- **KEY\_QUOTEDBL** = 34
- **KEY\_NUMBERSIGN** = 35
- **KEY\_DOLLAR** = 36
- **KEY\_PERCENT** = 37

- **KEY\_AMPERSAND** = 38
- **KEY\_APOSTROPHE** = 39
- **KEY\_PARENLEFT** = 40
- **KEY\_PARENRIGHT** = 41
- **KEY\_ASTERISK** = 42
- **KEY\_PLUS** = 43
- **KEY\_COMMA** = 44
- **KEY\_MINUS** = 45
- **KEY\_PERIOD** = 46
- **KEY\_SLASH** = 47
- **KEY\_0** = 48
- **KEY\_1** = 49
- **KEY\_2** = 50
- **KEY\_3** = 51
- **KEY\_4** = 52
- **KEY\_5** = 53
- **KEY\_6** = 54
- **KEY\_7** = 55
- **KEY\_8** = 56
- **KEY\_9** = 57
- **KEY\_COLON** = 58
- **KEY\_SEMICOLON** = 59
- **KEY\_LESS** = 60
- **KEY\_EQUAL** = 61
- **KEY\_GREATER** = 62
- **KEY\_QUESTION** = 63
- **KEY\_AT** = 64
- **KEY\_A** = 65
- **KEY\_B** = 66
- **KEY\_C** = 67
- **KEY\_D** = 68
- **KEY\_E** = 69
- **KEY\_F** = 70
- **KEY\_G** = 71
- **KEY\_H** = 72
- **KEY\_I** = 73

- **KEY\_J = 74**
- **KEY\_K = 75**
- **KEY\_L = 76**
- **KEY\_M = 77**
- **KEY\_N = 78**
- **KEY\_O = 79**
- **KEY\_P = 80**
- **KEY\_Q = 81**
- **KEY\_R = 82**
- **KEY\_S = 83**
- **KEY\_T = 84**
- **KEY\_U = 85**
- **KEY\_V = 86**
- **KEY\_W = 87**
- **KEY\_X = 88**
- **KEY\_Y = 89**
- **KEY\_Z = 90**
- **KEY\_BRACKETLEFT = 91**
- **KEY\_BACKSLASH = 92**
- **KEY\_BRACKETRIGHT = 93**
- **KEY\_ASCIICIRCUM = 94**
- **KEY\_UNDERSCORE = 95**
- **KEY\_QUOTELEFT = 96**
- **KEY\_BRACELEFT = 123**
- **KEY\_BAR = 124**
- **KEY\_BRACERIGHT = 125**
- **KEY\_ASCIITILDE = 126**
- **KEY\_NOBREAKSPACE = 160**
- **KEY\_EXCLAMDOWN = 161**
- **KEY\_CENT = 162**
- **KEY\_STERLING = 163**
- **KEY\_CURRENCY = 164**
- **KEY\_YEN = 165**
- **KEY\_BROKENBAR = 166**
- **KEY\_SECTION = 167**
- **KEY\_DIAERESIS = 168**

- **KEY\_COPYRIGHT = 169**
- **KEY\_ORDFEMININE = 170**
- **KEY\_GUILLEMOTLEFT = 171**
- **KEY\_NOTSIGN = 172**
- **KEY\_HYPHEN = 173**
- **KEY\_REGISTERED = 174**
- **KEY\_MACRON = 175**
- **KEY\_DEGREE = 176**
- **KEY\_PLUSMINUS = 177**
- **KEY\_TWOSUPERIOR = 178**
- **KEY\_THREESUPERIOR = 179**
- **KEY\_ACUTE = 180**
- **KEY\_MU = 181**
- **KEY\_PARAGRAPH = 182**
- **KEY\_PERIODCENTERED = 183**
- **KEY\_CEDILLA = 184**
- **KEY\_ONESUPERIOR = 185**
- **KEY\_MASCULINE = 186**
- **KEY\_GUILLEMOTRIGHT = 187**
- **KEY\_ONEQUARTER = 188**
- **KEY\_ONEHALF = 189**
- **KEY\_THREEQUARTERS = 190**
- **KEY\_QUESTIONDOWN = 191**
- **KEY\_AGRAVE = 192**
- **KEY\_AACUTE = 193**
- **KEY\_ACIRCUMFLEX = 194**
- **KEY\_ATILDE = 195**
- **KEY\_ADIAERESIS = 196**
- **KEY\_ARING = 197**
- **KEY\_AE = 198**
- **KEY\_CCEDILLA = 199**
- **KEY\_EGRAVE = 200**
- **KEY\_EACUTE = 201**
- **KEY\_ECIRCUMFLEX = 202**
- **KEY\_EDIAERESIS = 203**
- **KEY\_IGRAVE = 204**

- **KEY\_IACUTE** = 205
- **KEY\_ICIRCUMFLEX** = 206
- **KEY\_IDIAERESIS** = 207
- **KEY\_ETH** = 208
- **KEY\_NTILDE** = 209
- **KEY\_OGRAVE** = 210
- **KEY\_OACUTE** = 211
- **KEY\_OCIRCUMFLEX** = 212
- **KEY\_OTILDE** = 213
- **KEY\_ODIAERESIS** = 214
- **KEY\_MULTIPLY** = 215
- **KEY\_OOBLIQUE** = 216
- **KEY\_UGRAVE** = 217
- **KEY\_UACUTE** = 218
- **KEY\_UCIRCUMFLEX** = 219
- **KEY\_UDIAERESIS** = 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** = 33554432
- **KEY\_MASK\_ALT** = 67108864
- **KEY\_MASK\_META** = 134217728
- **KEY\_MASK\_CTRL** = 268435456
- **KEY\_MASK\_CMD** = 268435456
- **KEY\_MASK\_KPAD** = 536870912
- **KEY\_MASK\_GROUP\_SWITCH** = 1073741824
- **BUTTON\_LEFT** = 1
- **BUTTON\_RIGHT** = 2
- **BUTTON\_MIDDLE** = 3
- **BUTTON\_WHEEL\_UP** = 4
- **BUTTON\_WHEEL\_DOWN** = 5
- **BUTTON\_WHEEL\_LEFT** = 6

- **BUTTON\_WHEEL\_RIGHT = 7**
- **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**
- **JOY\_SNES\_B = 0**
- **JOY\_SNES\_X = 3**
- **JOY\_SNES\_Y = 2**
- **JOY\_SONY\_CIRCLE = 1**
- **JOY\_SONY\_X = 0**
- **JOY\_SONY\_SQUARE = 2**
- **JOY\_SONY\_TRIANGLE = 3**
- **JOY\_SEGA\_B = 1**
- **JOY\_SEGA\_A = 0**
- **JOY\_SEGA\_X = 2**
- **JOY\_SEGA\_Y = 3**
- **JOY\_XBOX\_B = 1**
- **JOY\_XBOX\_A = 0**
- **JOY\_XBOX\_X = 2**

- **JOY\_XBOX\_Y = 3**
- **JOY\_DS\_A = 1**
- **JOY\_DS\_B = 0**
- **JOY\_DS\_X = 3**
- **JOY\_DS\_Y = 2**
- **JOY\_SELECT = 10**
- **JOY\_START = 11**
- **JOY\_DPAD\_UP = 12**
- **JOY\_DPAD\_DOWN = 13**
- **JOY\_DPAD\_LEFT = 14**
- **JOY\_DPAD\_RIGHT = 15**
- **JOY\_L = 4**
- **JOY\_L2 = 6**
- **JOY\_L3 = 8**
- **JOY\_R = 5**
- **JOY\_R2 = 7**
- **JOY\_R3 = 9**
- **JOY\_AXIS\_0 = 0**
- **JOY\_AXIS\_1 = 1**
- **JOY\_AXIS\_2 = 2**
- **JOY\_AXIS\_3 = 3**
- **JOY\_AXIS\_4 = 4**
- **JOY\_AXIS\_5 = 5**
- **JOY\_AXIS\_6 = 6**
- **JOY\_AXIS\_7 = 7**
- **JOY\_AXIS\_MAX = 8**
- **JOY\_ANALOG\_0\_X = 0**
- **JOY\_ANALOG\_0\_Y = 1**
- **JOY\_ANALOG\_1\_X = 2**
- **JOY\_ANALOG\_1\_Y = 3**
- **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**
- **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\_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\_DEFAULT = 7** — Default usage (storage and editor).
- **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 integer.
- **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**

## 9.2.4 Description

Global scope constants and variables. This is all that resides in the globals, constants regarding error codes, scancodes, property hints, etc. It's not much.

Singletons are also documented here, since they can be accessed from anywhere.

## 9.3 AABB

**Category:** Built-In Types

### 9.3.1 Brief Description

Axis-Aligned Bounding Box.

### 9.3.2 Member Functions

<i>bool</i>	<code>encloses (AABB with )</code>
<i>AABB</i>	<code>expand ( Vector3 to_point )</code>
<i>float</i>	<code>get_area ()</code>
<i>Vector3</i>	<code>get_endpoint ( int idx )</code>
<i>Vector3</i>	<code>get_longest_axis ()</code>
<i>int</i>	<code>get_longest_axis_index ()</code>
<i>float</i>	<code>get_longest_axis_size ()</code>
<i>Vector3</i>	<code>get_shortest_axis ()</code>
<i>int</i>	<code>get_shortest_axis_index ()</code>
<i>float</i>	<code>get_shortest_axis_size ()</code>
<i>Vector3</i>	<code>get_support ( Vector3 dir )</code>
<i>AABB</i>	<code>grow ( float by )</code>
<i>bool</i>	<code>has_no_area ()</code>
<i>bool</i>	<code>has_no_surface ()</code>
<i>bool</i>	<code>has_point ( Vector3 point )</code>
<i>AABB</i>	<code>intersection ( AABB with )</code>
<i>bool</i>	<code>intersects ( AABB with )</code>
<i>bool</i>	<code>intersects_plane ( Plane plane )</code>
<i>bool</i>	<code>intersects_segment ( Vector3 from, Vector3 to )</code>
<i>AABB</i>	<code>merge ( AABB with )</code>
<i>AABB</i>	<code>AABB ( Vector3 pos, Vector3 size )</code>

### 9.3.3 Member Variables

- *Vector3 pos*
- *Vector3 size*
- *Vector3 end*

### 9.3.4 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.

### 9.3.5 Member Function Description

- `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)
- *AABB* **merge** (*AABB* with)

Combine this *AABB* with another, a larger one is returned that contains both.

- *AABB* **AABB** (*Vector3* pos, *Vector3* size)

Optional constructor, accepts position and size.

## 9.4 AcceptDialog

**Inherits:** *WindowDialog* < *Popup* < *Control* < *CanvasItem* < *Node* < *Object*

**Inherited By:** *ConfirmationDialog*

**Category:** Core

### 9.4.1 Brief Description

Base dialog for user notification.

### 9.4.2 Member Functions

<i>Object</i>	<i>get_ok</i> ()
<i>Object</i>	<i>get_label</i> ()
<i>void</i>	<i>set_hide_on_ok</i> ( <i>bool</i> enabled)
<i>bool</i>	<i>get_hide_on_ok</i> () const
<i>Button</i>	<i>add_button</i> ( <i>String</i> text, <i>bool</i> right=false, <i>String</i> action="")
<i>Button</i>	<i>add_cancel</i> ( <i>String</i> name)
<i>LineEdit</i>	<i>register_text_enter</i> ( <i>Object</i> line_edit)
<i>void</i>	<i>set_text</i> ( <i>String</i> text)
<i>String</i>	<i>get_text</i> () const

### 9.4.3 Signals

- **confirmed** ()
- **custom\_action** (*String* action)

### 9.4.4 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.

## 9.4.5 Member Function Description

- *Object* **get\_ok ()**

Return the OK Button.

- *Object* **get\_label ()**

Return the label used for built-in text.

- *void* **set\_hide\_on\_ok ( bool enabled )**

Set whether the dialog is hidden when accepted (default true).

- *bool* **get\_hide\_on\_ok () const**

Return true if the dialog will be hidden when accepted (default true).

- *Button* **add\_button ( String text, bool right=false, String action="" )**

- *Button* **add\_cancel ( String name )**

- *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\_text ( String text )**

Set the built-in label text.

- *String* **get\_text () const**

Return the built-in label text.

## 9.5 AnimatedSprite

**Inherits:** *Node2D* < *CanvasItem* < *Node* < *Object*

**Category:** Core

### 9.5.1 Brief Description

Sprite node that can use multiple textures for animation.

## 9.5.2 Member Functions

void	<code>set_sprite_frames ( <i>SpriteFrames</i> sprite_frames )</code>
<i>SpriteFrames</i>	<code>get_sprite_frames ( ) const</code>
void	<code>set_centered ( <i>bool</i> centered )</code>
<i>bool</i>	<code>is_centered ( ) const</code>
void	<code>set_offset ( <i>Vector2</i> offset )</code>
<i>Vector2</i>	<code>get_offset ( ) const</code>
void	<code>set_flip_h ( <i>bool</i> flip_h )</code>
<i>bool</i>	<code>is_flipped_h ( ) const</code>
void	<code>set_flip_v ( <i>bool</i> flip_v )</code>
<i>bool</i>	<code>is_flipped_v ( ) const</code>
void	<code>set_frame ( <i>int</i> frame )</code>
<i>int</i>	<code>get_frame ( ) const</code>
void	<code>set_modulate ( <i>Color</i> modulate )</code>
<i>Color</i>	<code>get_modulate ( ) const</code>

## 9.5.3 Signals

- `frame_changed ()`

## 9.5.4 Description

Sprite node that can use multiple textures for animation.

## 9.5.5 Member Function Description

- void `set_sprite_frames ( SpriteFrames sprite_frames )`

Set the *SpriteFrames* resource, which contains all frames.

- *SpriteFrames* `get_sprite_frames ( ) const`

Get the *SpriteFrames* resource, which contains all frames.

- 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.

- *bool* `is_centered ( ) const`

Return true when centered. See `set_centered`.

- 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.

- *Vector2* `get_offset ( ) const`

Return the offset of the sprite in the node origin.

- void `set_flip_h ( bool flip_h )`

If true, sprite is flipped horizontally.

- *bool* `is_flipped_h ( ) const`

Return true if sprite is flipped horizontally.

- void **set\_flip\_v** (*bool* flip\_v)

If true, sprite is flipped vertically.

- *bool* **is\_flipped\_v** () const

Return true if sprite is flipped vertically.

- void **set\_frame** (*int* frame)

Set the visible sprite frame index (from the list of frames inside the *SpriteFrames* resource).

- *int* **get\_frame** () const

Return the visible frame index.

- void **set\_modulate** (*Color* modulate)

Change the color modulation (multiplication) for this sprite.

- *Color* **get\_modulate** () const

Return the color modulation for this sprite.

## 9.6 AnimatedSprite3D

**Inherits:** *SpriteBase3D* < *GeometryInstance* < *VisualInstance* < *Spatial* < *Node* < *Object*

**Category:** Core

### 9.6.1 Brief Description

### 9.6.2 Member Functions

void	<i>set_sprite_frames</i> ( <i>SpriteFrames</i> sprite_frames)
<i>Texture</i>	<i>get_sprite_frames</i> () const
void	<i>set_frame</i> ( <i>int</i> frame)
<i>int</i>	<i>get_frame</i> () const

### 9.6.3 Signals

- **frame\_changed** ()

### 9.6.4 Member Function Description

- void **set\_sprite\_frames** (*SpriteFrames* sprite\_frames)
- *Texture* **get\_sprite\_frames** () const
- void **set\_frame** (*int* frame)
- *int* **get\_frame** () const

## 9.7 Animation

Inherits: [Resource](#) < [Reference](#) < [Object](#)

Category: Core

### 9.7.1 Brief Description

Contains data used to animate everything in the engine.

### 9.7.2 Member Functions

<i>int</i>	<code>add_track ( <i>int</i> type, <i>int</i> at_pos=-1 )</code>
<i>void</i>	<code>remove_track ( <i>int</i> idx )</code>
<i>int</i>	<code>get_track_count () const</code>
<i>int</i>	<code>track_get_type ( <i>int</i> idx ) const</code>
<i>NodePath</i>	<code>track_get_path ( <i>int</i> idx ) const</code>
<i>void</i>	<code>track_set_path ( <i>int</i> idx, <i>NodePath</i> path )</code>
<i>int</i>	<code>find_track ( <i>NodePath</i> path ) const</code>
<i>void</i>	<code>track_move_up ( <i>int</i> idx )</code>
<i>void</i>	<code>track_move_down ( <i>int</i> idx )</code>
<i>int</i>	<code>transform_track_insert_key ( <i>int</i> idx, <i>float</i> time, <i>Vector3</i> loc, <i>Quat</i> rot, <i>Vector3</i> scale )</code>
<i>void</i>	<code>track_insert_key ( <i>int</i> idx, <i>float</i> time, var key, <i>float</i> transition=1 )</code>
<i>void</i>	<code>track_remove_key ( <i>int</i> idx, <i>int</i> key_idx )</code>
<i>void</i>	<code>track_remove_key_at_pos ( <i>int</i> idx, <i>float</i> pos )</code>
<i>void</i>	<code>track_set_key_value ( <i>int</i> idx, <i>int</i> key, var value )</code>
<i>void</i>	<code>track_set_key_transition ( <i>int</i> idx, <i>int</i> key_idx, <i>float</i> transition )</code>
<i>float</i>	<code>track_get_key_transition ( <i>int</i> idx, <i>int</i> key_idx ) const</code>
<i>int</i>	<code>track_get_key_count ( <i>int</i> idx ) const</code>
<i>void</i>	<code>track_get_key_value ( <i>int</i> idx, <i>int</i> key_idx ) const</code>
<i>float</i>	<code>track_get_key_time ( <i>int</i> idx, <i>int</i> key_idx ) const</code>
<i>int</i>	<code>track_find_key ( <i>int</i> idx, <i>float</i> time, <i>bool</i> exact=false ) const</code>
<i>void</i>	<code>track_set_interpolation_type ( <i>int</i> idx, <i>int</i> interpolation )</code>
<i>int</i>	<code>track_get_interpolation_type ( <i>int</i> idx ) const</code>
<i>Array</i>	<code>transform_track_interpolate ( <i>int</i> idx, <i>float</i> time_sec ) const</code>
<i>void</i>	<code>value_track_set_continuous ( <i>int</i> idx, <i>bool</i> continuous )</code>
<i>bool</i>	<code>value_track_is_continuous ( <i>int</i> idx ) const</code>
<i>IntArray</i>	<code>value_track_get_key_indices ( <i>int</i> idx, <i>float</i> time_sec, <i>float</i> delta ) const</code>
<i>IntArray</i>	<code>method_track_get_key_indices ( <i>int</i> idx, <i>float</i> time_sec, <i>float</i> delta ) const</code>
<i>String</i>	<code>method_track_get_name ( <i>int</i> idx, <i>int</i> key_idx ) const</code>
<i>Array</i>	<code>method_track_get_params ( <i>int</i> idx, <i>int</i> key_idx ) const</code>
<i>void</i>	<code>set_length ( <i>float</i> time_sec )</code>
<i>float</i>	<code>get_length () const</code>
<i>void</i>	<code>set_loop ( <i>bool</i> enabled )</code>
<i>bool</i>	<code>has_loop () const</code>
<i>void</i>	<code>set_step ( <i>float</i> size_sec )</code>
<i>float</i>	<code>get_step () const</code>
<i>void</i>	<code>clear ()</code>

### 9.7.3 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.

### 9.7.4 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 nodes such as an *AnimationPlayer* or *AnimationTreePlayer* to be played back.

### 9.7.5 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 remove_track ( int idx )`

Remove a track by specifying the track index.

- `int get_track_count ( ) const`

Return the amount of tracks in the animation.

- `int track_get_type ( int idx ) const`

Get the type of a 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*

- `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 find_track ( NodePath path ) const`
- `void track_move_up ( int idx )`

Move a track up.

- `void track_move_down ( int idx )`

Move a track down.

- `int transform_track_insert_key ( int idx, float time, Vector3 loc, Quat rot, Vector3 scale )`

Insert a transform key for a transform track.

- `void track_insert_key ( int idx, float time, var key, float transition=1 )`

Insert a generic key in a given track.

- `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_key_value ( int idx, int key, var value )`

Set the value of an existing key.

- `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”).

- `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”).

- `int track_get_key_count ( int idx ) const`

Return the amount of keys in a given track.

- `void track_get_key_value ( int idx, int key_idx ) const`

Return the value of a given key in a given track.

- `float track_get_key_time ( int idx, int key_idx ) const`

Return the time at which the key is located.

- `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.

- `void track_set_interpolation_type ( int idx, int interpolation )`

Set the interpolation type of a given track, from the INTERPOLATION\_\* enum.

- `int track_get_interpolation_type ( int idx ) const`

Return the interpolation type of a given track, from the INTERPOLATION\_\* enum.

- `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](#)).

- `void value_track_set_continuous ( int idx, bool continuous )`

Enable or disable interpolation for a whole track. By default tracks are interpolated.

- `bool value_track_is_continuous ( int idx ) const`

Return whether interpolation is enabled or disabled for a whole track. By default tracks are interpolated.

- `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.

- `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 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.

- `float get_length () const`

Return the total length of the animation (in seconds).

- `void set_loop ( bool enabled )`

Set a flag indicating that the animation must loop. This is used for correct interpolation of animation cycles, and for hinting the player that it must restart the animation.

- `bool has_loop () const`

Return whether the animation has the loop flag set.

- `void set_step ( float size_sec )`
- `float get_step () const`
- `void clear ()`

Clear the animation (clear all tracks and reset all).

## 9.8 AnimationPlayer

**Inherits:** `Node < Object`

**Category:** Core

### 9.8.1 Brief Description

Container and player of *Animation* resources.

### 9.8.2 Member Functions

<code>int</code>	<code>add_animation ( String name, Animation animation )</code>
<code>void</code>	<code>remove_animation ( String name )</code>
<code>void</code>	<code>rename_animation ( String name, String newname )</code>
<code>bool</code>	<code>has_animation ( String name ) const</code>
<code>Animation</code>	<code>get_animation ( String name ) const</code>
<code>StringArray</code>	<code>get_animation_list () const</code>
<code>void</code>	<code>set_blend_time ( String anim_from, String anim_to, float sec )</code>
<code>float</code>	<code>get_blend_time ( String anim_from, String anim_to ) const</code>
<code>void</code>	<code>set_default_blend_time ( float sec )</code>
<code>float</code>	<code>get_default_blend_time () const</code>

Continued on next page

Table 9.3 – continued from previous page

void	<code>play ( String name="" , float custom_blend=-1 , float custom_speed=1 , bool from_end=false )</code>
void	<code>play_backwards ( String name="" , float custom_blend=-1 )</code>
void	<code>stop ( bool reset=true )</code>
void	<code>stop_all ()</code>
<code>bool</code>	<code>is_playing () const</code>
void	<code>set_current_animation ( String anim )</code>
<code>String</code>	<code>get_current_animation () const</code>
void	<code>queue ( String name )</code>
void	<code>clear_queue ()</code>
void	<code>set_active ( bool active )</code>
<code>bool</code>	<code>is_active () const</code>
void	<code>set_speed ( float speed )</code>
<code>float</code>	<code>get_speed () const</code>
void	<code>set_autoplay ( String name )</code>
<code>String</code>	<code>get_autoplay () const</code>
void	<code>set_root ( NodePath path )</code>
<code>NodePath</code>	<code>get_root () const</code>
void	<code>seek ( float pos_sec , bool update=false )</code>
<code>float</code>	<code>get_pos () const</code>
<code>String</code>	<code>find_animation ( Animation animation ) const</code>
void	<code>clear_caches ()</code>
void	<code>set_animation_process_mode ( int mode )</code>
<code>int</code>	<code>get_animation_process_mode () const</code>
<code>float</code>	<code>get_current_animation_pos () const</code>
<code>float</code>	<code>get_current_animation_length () const</code>
void	<code>advance ( float delta )</code>

### 9.8.3 Signals

- `animation_changed ( String old_name , String new_name )`
- `finished ()`

### 9.8.4 Numeric Constants

- **ANIMATION\_PROCESS\_FIXED = 0** — Process animation on fixed process. This is specially useful when animating kinematic bodies.
- **ANIMATION\_PROCESS\_IDLE = 1** — Process animation on idle process.

### 9.8.5 Description

An animation player is used for general purpose playback of `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.

### 9.8.6 Member Function Description

- `int add_animation ( String name , Animation animation )`

Add an animation resource to the player, which will be later referenced by the “name” argument.

- `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.

- `bool has_animation ( String name ) const`

Request whether an *Animation* name exist within the player.

- `Animation get_animation ( String name ) const`

Get an *Animation* resource by requesting a name.

- `StringArray get_animation_list ( ) const`

Get the list of names of the animations stored in the player.

- `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.

- `float get_blend_time ( String anim_from, String anim_to ) const`

Get the blend time between two animations, referenced by their names.

- `void set_default_blend_time ( float sec )`

Set the default blend time between animations.

- `float get_default_blend_time ( ) const`

Return the default blend time between animations.

- `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 )`

- `void stop ( bool reset=true )`

- `void stop_all ( )`

Stop playback of animations (deprecated).

- `bool is_playing ( ) const`

Return whether an animation is playing.

- `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()`.

- `String get_current_animation ( ) const`

Return the name of the animation being played.

- `void queue ( String name )`

Queue an animation for playback once the current one is done.

- `void clear_queue ( )`

If animations are queued to play, clear them.

- `void set_active ( bool active )`

Set the player as active (playing). If false, it will do nothing.

- `bool is_active () const`

Return true if the player is active.

- `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).

- `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).

- `void set_autoplay ( String name )`

Set the name of the animation that will be automatically played when the scene is loaded.

- `String get_autoplay () const`

Return the name of the animation that will be automatically played when the scene is loaded.

- `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.

- `NodePath get_root () const`

Return path to root node (see `set_root`).

- `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.

- `float get_pos () const`

Return the playback position (in seconds) in an animation channel (or channel 0 if none is provided).

- `String find_animation ( Animation animation ) const`

Find an animation name by resource.

- `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.

- `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_*`.

- `int get_animation_process_mode () const`

Return the mode in which the animation player processes. See `set_animation_process_mode`.

- `float get_current_animation_pos () const`

Get the position (in seconds) of the currently being played animation.

- `float get_current_animation_length () const`

Get the length (in seconds) of the currently being played animation.

- `void advance (float delta )`

## 9.9 AnimationTreePlayer

**Inherits:** `Node < Object`

**Category:** Core

### 9.9.1 Brief Description

Animation Player that uses a node graph for the blending.

### 9.9.2 Member Functions

<code>void</code>	<code>add_node ( int type, String id )</code>
<code>bool</code>	<code>node_exists ( String node ) const</code>
<code>int</code>	<code>node_rename ( String node, String new_name )</code>
<code>int</code>	<code>node_get_type ( String id ) const</code>
<code>int</code>	<code>node_get_input_count ( String id ) const</code>
<code>String</code>	<code>node_get_input_source ( String id, int idx ) const</code>
<code>void</code>	<code>animation_node_set_animation ( String id, Animation animation )</code>
<code>Animation</code>	<code>animation_node_get_animation ( String id ) const</code>
<code>void</code>	<code>animation_node_set_master_animation ( String id, String source )</code>
<code>String</code>	<code>animation_node_get_master_animation ( String id ) const</code>
<code>void</code>	<code>oneshot_node_set_fadein_time ( String id, float time_sec )</code>
<code>float</code>	<code>oneshot_node_get_fadein_time ( String id ) const</code>
<code>void</code>	<code>oneshot_node_set_fadeout_time ( String id, float time_sec )</code>
<code>float</code>	<code>oneshot_node_get_fadeout_time ( String id ) const</code>
<code>void</code>	<code>oneshot_node_set_autorestart ( String id, bool enable )</code>
<code>void</code>	<code>oneshot_node_set_autorestart_delay ( String id, float delay_sec )</code>
<code>void</code>	<code>oneshot_node_set_autorestart_random_delay ( String id, float rand_sec )</code>
<code>bool</code>	<code>oneshot_node_has_autorestart ( String id ) const</code>
<code>float</code>	<code>oneshot_node_get_autorestart_delay ( String id ) const</code>
<code>float</code>	<code>oneshot_node_get_autorestart_random_delay ( String id ) const</code>
<code>void</code>	<code>oneshot_node_start ( String id )</code>
<code>void</code>	<code>oneshot_node_stop ( String id )</code>
<code>bool</code>	<code>oneshot_node_is_active ( String id ) const</code>
<code>void</code>	<code>oneshot_node_set_filter_path ( String id, NodePath path, bool enable )</code>
<code>void</code>	<code>mix_node_set_amount ( String id, float ratio )</code>
<code>float</code>	<code>mix_node_get_amount ( String id ) const</code>
<code>void</code>	<code>blend2_node_set_amount ( String id, float blend )</code>
<code>float</code>	<code>blend2_node_get_amount ( String id ) const</code>
<code>void</code>	<code>blend2_node_set_filter_path ( String id, NodePath path, bool enable )</code>
<code>void</code>	<code>blend3_node_set_amount ( String id, float blend )</code>
<code>float</code>	<code>blend3_node_get_amount ( String id ) const</code>

Continued on next page

Table 9.4 – continued from previous page

void	<i>blend4_node_set_amount</i> ( <i>String id</i> , <i>Vector2 blend</i> )
<i>Vector2</i>	<i>blend4_node_get_amount</i> ( <i>String id</i> ) const
void	<i>timescale_node_set_scale</i> ( <i>String id</i> , <i>float scale</i> )
<i>float</i>	<i>timescale_node_get_scale</i> ( <i>String id</i> ) const
void	<i>timeseek_node_seek</i> ( <i>String id</i> , <i>float pos_sec</i> )
void	<i>transition_node_set_input_count</i> ( <i>String id</i> , <i>int count</i> )
<i>int</i>	<i>transition_node_get_input_count</i> ( <i>String id</i> ) const
void	<i>transition_node_delete_input</i> ( <i>String id</i> , <i>int input_idx</i> )
void	<i>transition_node_set_input_auto_advance</i> ( <i>String id</i> , <i>int input_idx</i> , <i>bool enable</i> )
<i>bool</i>	<i>transition_node_has_input_auto_advance</i> ( <i>String id</i> , <i>int input_idx</i> ) const
void	<i>transition_node_set_xfade_time</i> ( <i>String id</i> , <i>float time_sec</i> )
<i>float</i>	<i>transition_node_get_xfade_time</i> ( <i>String id</i> ) const
void	<i>transition_node_set_current</i> ( <i>String id</i> , <i>int input_idx</i> )
<i>int</i>	<i>transition_node_get_current</i> ( <i>String id</i> ) const
void	<i>node_set_pos</i> ( <i>String id</i> , <i>Vector2 screen_pos</i> )
<i>Vector2</i>	<i>node_get_pos</i> ( <i>String id</i> ) const
void	<i>remove_node</i> ( <i>String id</i> )
<i>int</i>	<i>connect</i> ( <i>String id</i> , <i>String dst_id</i> , <i>int dst_input_idx</i> )
<i>bool</i>	<i>is_connected</i> ( <i>String id</i> , <i>String dst_id</i> , <i>int dst_input_idx</i> ) const
void	<i>disconnect</i> ( <i>String id</i> , <i>int dst_input_idx</i> )
void	<i>set_active</i> ( <i>bool enabled</i> )
<i>bool</i>	<i>is_active</i> ( ) const
void	<i>set_base_path</i> ( <i>NodePath path</i> )
<i>NodePath</i>	<i>get_base_path</i> ( ) const
void	<i>set_master_player</i> ( <i>NodePath nodepath</i> )
<i>NodePath</i>	<i>get_master_player</i> ( ) const
<i>StringArray</i>	<i>get_node_list</i> ( )
void	<i>set_animation_process_mode</i> ( <i>int mode</i> )
<i>int</i>	<i>get_animation_process_mode</i> ( ) const
void	<i>advance</i> ( <i>float delta</i> )
void	<i>reset</i> ( )
void	<i>recompute_caches</i> ( )

### 9.9.3 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**

## 9.9.4 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.

## 9.9.5 Member Function Description

- `void add_node ( int type, String id )`

Add a node of a given type in the graph with given id.

- `bool node_exists ( String node ) const`

Check if a node exists (by name).

- `int node_rename ( String node, String new_name )`

Rename a node in the graph.

- `int node_get_type ( String id ) const`

Get the node type, will return from NODE\_\* enum.

- `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.

- `void animation_node_set_animation ( String id, Animation animation )`

Set the animation for an animation node.

- `Animation animation_node_get_animation ( String id ) const`

- `void animation_node_set_master_animation ( String id, String source )`

- `String animation_node_get_master_animation ( String id ) const`

- `void oneshot_node_set_fadein_time ( String id, float time_sec )`

- `float oneshot_node_get_fadein_time ( String id ) const`

- `void oneshot_node_set_fadeout_time ( String id, float time_sec )`

- `float oneshot_node_get_fadeout_time ( 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 )`

- `bool oneshot_node_has_autorestart ( String id ) const`

- `float oneshot_node_get_autorestart_delay ( String id ) const`

- `float oneshot_node_get_autorestart_random_delay ( String id ) const`

- `void oneshot_node_start ( String id )`

- `void oneshot_node_stop ( String id )`

- `bool oneshot_node_is_active ( String id ) const`

- `void oneshot_node_set_filter_path ( String id, NodePath path, bool enable )`

- void **mix\_node\_set\_amount** ( *String* id, *float* ratio )
- *float* **mix\_node\_get\_amount** ( *String* id ) const
- void **blend2\_node\_set\_amount** ( *String* id, *float* blend )
- *float* **blend2\_node\_get\_amount** ( *String* id ) const
- void **blend2\_node\_set\_filter\_path** ( *String* id, *NodePath* path, *bool* enable )
- void **blend3\_node\_set\_amount** ( *String* id, *float* blend )
- *float* **blend3\_node\_get\_amount** ( *String* id ) const
- void **blend4\_node\_set\_amount** ( *String* id, *Vector2* blend )
- *Vector2* **blend4\_node\_get\_amount** ( *String* id ) const
- void **timescale\_node\_set\_scale** ( *String* id, *float* scale )
- *float* **timescale\_node\_get\_scale** ( *String* id ) const
- void **timeseek\_node\_seek** ( *String* id, *float* pos\_sec )
- void **transition\_node\_set\_input\_count** ( *String* id, *int* count )
- *int* **transition\_node\_get\_input\_count** ( *String* id ) const
- void **transition\_node\_delete\_input** ( *String* id, *int* input\_idx )
- void **transition\_node\_set\_input\_auto\_advance** ( *String* id, *int* input\_idx, *bool* enable )
- *bool* **transition\_node\_has\_input\_auto\_advance** ( *String* id, *int* input\_idx ) const
- void **transition\_node\_set\_xfade\_time** ( *String* id, *float* time\_sec )
- *float* **transition\_node\_get\_xfade\_time** ( *String* id ) const
- void **transition\_node\_set\_current** ( *String* id, *int* input\_idx )
- *int* **transition\_node\_get\_current** ( *String* id ) const
- void **node\_set\_pos** ( *String* id, *Vector2* screen\_pos )
- *Vector2* **node\_get\_pos** ( *String* id ) const
- void **remove\_node** ( *String* id )
- *int* **connect** ( *String* id, *String* dst\_id, *int* dst\_input\_idx )
- *bool* **is\_connected** ( *String* id, *String* dst\_id, *int* dst\_input\_idx ) const
- void **disconnect** ( *String* id, *int* dst\_input\_idx )
- void **set\_active** ( *bool* enabled )
- *bool* **is\_active** ( ) const
- void **set\_base\_path** ( *NodePath* path )
- *NodePath* **get\_base\_path** ( ) const
- void **set\_master\_player** ( *NodePath* nodepath )
- *NodePath* **get\_master\_player** ( ) const
- *StringArray* **get\_node\_list** ( )
- void **set\_animation\_process\_mode** ( *int* mode )
- *int* **get\_animation\_process\_mode** ( ) const

- void **advance** (*float* delta)
- void **reset** ()
- void **recompute\_caches** ()

## 9.10 Area

**Inherits:** *CollisionObject* < *Spatial* < *Node* < *Object*

**Category:** Core

### 9.10.1 Brief Description

### 9.10.2 Member Functions

void	<i>set_space_override_mode</i> ( <i>int</i> enable)
<i>int</i>	<i>get_space_override_mode</i> () const
void	<i>set_gravity_is_point</i> ( <i>bool</i> enable)
<i>bool</i>	<i>is_gravity_a_point</i> () const
void	<i>set_gravity_distance_scale</i> ( <i>float</i> distance_scale)
<i>float</i>	<i>get_gravity_distance_scale</i> () const
void	<i>set_gravity_vector</i> ( <i>Vector3</i> vector)
<i>Vector3</i>	<i>get_gravity_vector</i> () const
void	<i>set_gravity</i> ( <i>float</i> gravity)
<i>float</i>	<i>get_gravity</i> () const
void	<i>set_angular_damp</i> ( <i>float</i> angular_damp)
<i>float</i>	<i>get_angular_damp</i> () const
void	<i>set_linear_damp</i> ( <i>float</i> linear_damp)
<i>float</i>	<i>get_linear_damp</i> () const
void	<i>set_priority</i> ( <i>float</i> priority)
<i>float</i>	<i>get_priority</i> () const
void	<i>set_monitorable</i> ( <i>bool</i> enable)
<i>bool</i>	<i>is_monitorable</i> () const
void	<i>set_enable_monitoring</i> ( <i>bool</i> enable)
<i>bool</i>	<i>is_monitoring_enabled</i> () const
<i>Array</i>	<i>get_overlapping_bodies</i> () const
<i>Array</i>	<i>get_overlapping_areas</i> () const
<i>PhysicsBody</i>	<i>overlaps_body</i> ( <i>Object</i> body) const
<i>Area</i>	<i>overlaps_area</i> ( <i>Object</i> area) const

### 9.10.3 Signals

- **body\_enter** (*Object* body)
- **body\_enter\_shape** (*int* body\_id, *Object* body, *int* body\_shape, *int* area\_shape)
- **area\_enter** (*Object* area)
- **area\_enter\_shape** (*int* area\_id, *Object* area, *int* area\_shape, *int* area\_shape)
- **body\_exit** (*Object* body)

- **body\_exit\_shape** ( *int* body\_id, *Object* body, *int* body\_shape, *int* area\_shape )
- **area\_exit** ( *Object* area )
- **area\_exit\_shape** ( *int* area\_id, *Object* area, *int* area\_shape, *int* area\_shape )

## 9.10.4 Member Function Description

- 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.

- *int* **get\_space\_override\_mode** () const
- void **set\_gravity\_is\_point** ( *bool* enable )
- *bool* **is\_gravity\_a\_point** () const
- void **set\_gravity\_distance\_scale** ( *float* distance\_scale )
- *float* **get\_gravity\_distance\_scale** () const
- void **set\_gravity\_vector** ( *Vector3* vector )
- *Vector3* **get\_gravity\_vector** () const
- void **set\_gravity** ( *float* gravity )
- *float* **get\_gravity** () const
- void **set\_angular\_damp** ( *float* angular\_damp )
- *float* **get\_angular\_damp** () const
- void **set\_linear\_damp** ( *float* linear\_damp )
- *float* **get\_linear\_damp** () const
- void **set\_priority** ( *float* priority )
- *float* **get\_priority** () const
- void **set\_monitorable** ( *bool* enable )
- *bool* **is\_monitorable** () const
- void **set\_enable\_monitoring** ( *bool* enable )
- *bool* **is\_monitoring\_enabled** () const
- *Array* **get\_overlapping\_bodies** () const
- *Array* **get\_overlapping\_areas** () const

- *PhysicsBody* `overlaps_body ( Object body ) const`
- *Area* `overlaps_area ( Object area ) const`

## 9.11 Area2D

**Inherits:** `CollisionObject2D < Node2D < CanvasItem < Node < Object`

**Category:** Core

### 9.11.1 Brief Description

General purpose area detection and influence for 2D physics.

### 9.11.2 Member Functions

<code>void</code>	<code>set_space_override_mode ( int enable )</code>
<code>int</code>	<code>get_space_override_mode () const</code>
<code>void</code>	<code>set_gravity_is_point ( bool enable )</code>
<code>bool</code>	<code>is_gravity_a_point () const</code>
<code>void</code>	<code>set_gravity_distance_scale ( float distance_scale )</code>
<code>float</code>	<code>get_gravity_distance_scale () const</code>
<code>void</code>	<code>set_gravity_vector ( Vector2 vector )</code>
<code>Vector2</code>	<code>get_gravity_vector () const</code>
<code>void</code>	<code>set_gravity ( float gravity )</code>
<code>float</code>	<code>get_gravity () const</code>
<code>void</code>	<code>set_linear_damp ( float linear_damp )</code>
<code>float</code>	<code>get_linear_damp () const</code>
<code>void</code>	<code>set_angular_damp ( float angular_damp )</code>
<code>float</code>	<code>get_angular_damp () const</code>
<code>void</code>	<code>set_priority ( float priority )</code>
<code>float</code>	<code>get_priority () const</code>
<code>void</code>	<code>set_collision_mask ( int collision_mask )</code>
<code>int</code>	<code>get_collision_mask () const</code>
<code>void</code>	<code>set_layer_mask ( int layer_mask )</code>
<code>int</code>	<code>get_layer_mask () const</code>
<code>void</code>	<code>set_collision_mask_bit ( int bit, bool value )</code>
<code>bool</code>	<code>get_collision_mask_bit ( int bit ) const</code>
<code>void</code>	<code>set_layer_mask_bit ( int bit, bool value )</code>
<code>bool</code>	<code>get_layer_mask_bit ( int bit ) const</code>
<code>void</code>	<code>set_enable_monitoring ( bool enable )</code>
<code>bool</code>	<code>is_monitoring_enabled () const</code>
<code>void</code>	<code>set_monitorable ( bool enable )</code>
<code>bool</code>	<code>is_monitorable () const</code>
<code>Array</code>	<code>get_overlapping_bodies () const</code>
<code>Array</code>	<code>get_overlapping_areas () const</code>
<code>PhysicsBody2D</code>	<code>overlaps_body ( Object body ) const</code>
<code>Area2D</code>	<code>overlaps_area ( Object area ) const</code>

### 9.11.3 Signals

- **body\_enter** ( *Object* body )
- **body\_enter\_shape** ( *int* body\_id, *Object* body, *int* body\_shape, *int* area\_shape )
- **area\_enter** ( *Object* area )
- **area\_enter\_shape** ( *int* area\_id, *Object* area, *int* area\_shape, *int* area\_shape )
- **body\_exit** ( *Object* body )
- **body\_exit\_shape** ( *int* body\_id, *Object* body, *int* body\_shape, *int* area\_shape )
- **area\_exit** ( *Object* area )
- **area\_exit\_shape** ( *int* area\_id, *Object* area, *int* area\_shape, *int* area\_shape )

### 9.11.4 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.

### 9.11.5 Member Function Description

- 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.

- *int* **get\_space\_override\_mode** ( ) const

Return the space override mode.

- 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](#).

- *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.

- 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.

- `float get_gravity_distance_scale () const`

Return the falloff factor for point gravity.

- `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.

- `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.

- `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).

- `float get_gravity () const`

Return the gravity intensity.

- `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.

- `float get_linear_damp () const`

Return the linear damp rate.

- `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.

- `float get_angular_damp () const`

Return the angular damp rate.

- `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.

- `float get_priority () const`

Return the processing order of this area.

- `void set_collision_mask ( int collision_mask )`

Set the physics layers this area can scan for collisions.

- `int get_collision_mask () const`

Return the physics layers this area can scan for collisions.

- `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.

- `int get_layer_mask () const`

Return the physics layer this area is in.

- `void set_collision_mask_bit ( int bit, bool value )`

Set/clear individual bits on the collision mask. This makes selecting the areas scanned easier.

- `bool get_collision_mask_bit ( int bit ) const`

Return an individual bit on the collision mask.

- `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.

- `bool get_layer_mask_bit ( int bit ) const`

Return an individual bit on the layer mask.

- `void set_enable_monitoring ( bool enable )`

Set whether this area can detect bodies/areas entering/exiting it.

- `bool is_monitoring_enabled () const`

Return whether this area detects bodies/areas entering/exiting it.

- `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.

- `bool is_monitorable () const`

Set whether this area can be detected by other, monitoring, areas.

- `Array<PhysicsBody2D> get_overlapping_bodies () const`

Return a list of the bodies (`PhysicsBody2D`) that are totally or partially inside this area.

- `Array<Area2D> get_overlapping_areas () const`

Return a list of the areas that are totally or partially inside this area.

- `PhysicsBody2D overlaps_body ( Object body ) const`

Return whether the body passed is totally or partially inside this area.

- `Area2D overlaps_area ( Object area ) const`

Return whether the area passed is totally or partially inside this area.

## 9.12 Array

**Category:** Built-In Types

### 9.12.1 Brief Description

Generic array datatype.

### 9.12.2 Member Functions

void	<i>append</i> ( var value )
void	<i>clear</i> ( )
<i>bool</i>	<i>empty</i> ( )
void	<i>erase</i> ( var value )
<i>int</i>	<i>find</i> ( var value )
<i>int</i>	<i>hash</i> ( )
void	<i>insert</i> ( <i>int</i> pos, var value )
void	<i>invert</i> ( )
<i>bool</i>	<i>is_shared</i> ( )
void	<i>pop_back</i> ( )
void	<i>pop_front</i> ( )
void	<i>push_back</i> ( var value )
void	<i>push_front</i> ( var value )
void	<i>remove</i> ( <i>int</i> pos )
void	<i>resize</i> ( <i>int</i> pos )
<i>int</i>	<i>size</i> ( )
void	<i>sort</i> ( )
void	<i>sort_custom</i> ( <i>Object</i> obj, <i>String</i> func )
<i>Array</i>	Array ( <i>RawArray</i> from )
<i>Array</i>	Array ( <i>IntArray</i> from )
<i>Array</i>	Array ( <i>RealArray</i> from )
<i>Array</i>	Array ( <i>StringArray</i> from )
<i>Array</i>	Array ( <i>Vector2Array</i> from )
<i>Array</i>	Array ( <i>Vector3Array</i> from )
<i>Array</i>	Array ( <i>ColorArray</i> from )

### 9.12.3 Description

Generic array, contains several elements of any type, accessible by numerical index starting at 0. Arrays are always passed by reference.

### 9.12.4 Member Function Description

- void **append** ( var value )

Append an element at the end of the array (alias of *push\_back*).

- void **clear** ( )

Clear the array (resize to 0).

- `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 value )`

Searches the array for a value and returns its index or -1 if not found.

- `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()`

- `void pop_front()`

- `void push_back( var value )`

Append an element at the end of the array.

- `void push_front( var value )`

- `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 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.

- `Array Array( RawArray from )`

Construct an array from a `RawArray`.

- `Array Array( IntArray from )`

Construct an array from a `RawArray`.

- [Array Array \( `RealArray` from \)](#)

Construct an array from a `RawArray`.

- [Array Array \( `StringArray` from \)](#)

Construct an array from a `RawArray`.

- [Array Array \( `Vector2Array` from \)](#)

Construct an array from a `RawArray`.

- [Array Array \( `Vector3Array` from \)](#)

Construct an array from a `RawArray`.

- [Array Array \( `ColorArray` from \)](#)

Construct an array from a `RawArray`.

## 9.13 AtlasTexture

**Inherits:** `Texture < Resource < Reference < Object`

**Category:** Core

### 9.13.1 Brief Description

### 9.13.2 Member Functions

void	<a href="#"><code>set_atlas ( Texture atlas )</code></a>
<code>Texture</code>	<a href="#"><code>get_atlas ( ) const</code></a>
void	<a href="#"><code>set_region ( Rect2 region )</code></a>
<code>Rect2</code>	<a href="#"><code>get_region ( ) const</code></a>
void	<a href="#"><code>set_margin ( Rect2 margin )</code></a>
<code>Rect2</code>	<a href="#"><code>get_margin ( ) const</code></a>

### 9.13.3 Member Function Description

- void [set\\_atlas \( `Texture` atlas \)](#)
- `Texture` [get\\_atlas \( \) const](#)
- void [set\\_region \( `Rect2` region \)](#)
- `Rect2` [get\\_region \( \) const](#)
- void [set\\_margin \( `Rect2` margin \)](#)
- `Rect2` [get\\_margin \( \) const](#)

## 9.14 AudioServer

**Inherits:** `Object`

**Inherited By:** `AudioServerSW`

Category: Core

### 9.14.1 Brief Description

Server interface for low level audio access.

### 9.14.2 Member Functions

<i>RID</i>	<i>sample_create</i> ( <i>int</i> format, <i>bool</i> stereo, <i>int</i> length )
<i>void</i>	<i>sample_set_description</i> ( <i>RID</i> sample, <i>String</i> description )
<i>String</i>	<i>sample_get_description</i> ( <i>RID</i> sample ) const
<i>int</i>	<i>sample_get_format</i> ( <i>RID</i> sample ) const
<i>bool</i>	<i>sample_is_stereo</i> ( <i>RID</i> sample ) const
<i>int</i>	<i>sample_get_length</i> ( <i>RID</i> sample ) const
<i>void</i>	<i>sample_set_signed_data</i> ( <i>RID</i> sample, <i>RealArray</i> data )
<i>void</i>	<i>sample_set_data</i> ( <i>RID</i> sample, <i>RawArray</i> data )
<i>RawArray</i>	<i>sample_get_data</i> ( <i>RID</i> sample ) const
<i>void</i>	<i>sample_set_mix_rate</i> ( <i>RID</i> sample, <i>int</i> mix_rate )
<i>int</i>	<i>sample_get_mix_rate</i> ( <i>RID</i> sample ) const
<i>void</i>	<i>sample_set_loop_format</i> ( <i>RID</i> sample, <i>int</i> loop_format )
<i>int</i>	<i>sample_get_loop_format</i> ( <i>RID</i> sample ) const
<i>void</i>	<i>sample_set_loop_begin</i> ( <i>RID</i> sample, <i>int</i> pos )
<i>int</i>	<i>sample_get_loop_begin</i> ( <i>RID</i> sample ) const
<i>void</i>	<i>sample_set_loop_end</i> ( <i>RID</i> sample, <i>int</i> pos )
<i>int</i>	<i>sample_get_loop_end</i> ( <i>RID</i> sample ) const
<i>RID</i>	<i>voice_create</i> ( )
<i>void</i>	<i>voice_play</i> ( <i>RID</i> voice, <i>RID</i> sample )
<i>void</i>	<i>voice_set_volume</i> ( <i>RID</i> voice, <i>float</i> volume )
<i>void</i>	<i>voice_set_pan</i> ( <i>RID</i> voice, <i>float</i> pan, <i>float</i> depth=0, <i>float</i> height=0 )
<i>void</i>	<i>voice_set_filter</i> ( <i>RID</i> voice, <i>int</i> type, <i>float</i> cutoff, <i>float</i> resonance, <i>float</i> gain=0 )
<i>void</i>	<i>voice_set_chorus</i> ( <i>RID</i> voice, <i>float</i> chorus )
<i>void</i>	<i>voice_set_reverb</i> ( <i>RID</i> voice, <i>int</i> room, <i>float</i> reverb )
<i>void</i>	<i>voice_set_mix_rate</i> ( <i>RID</i> voice, <i>int</i> rate )
<i>void</i>	<i>voice_set_positional</i> ( <i>RID</i> voice, <i>bool</i> enabled )
<i>float</i>	<i>voice_get_volume</i> ( <i>RID</i> voice ) const
<i>float</i>	<i>voice_get_pan</i> ( <i>RID</i> voice ) const
<i>float</i>	<i>voice_get_pan_height</i> ( <i>RID</i> voice ) const
<i>float</i>	<i>voice_get_pan_depth</i> ( <i>RID</i> voice ) const
<i>int</i>	<i>voice_get_filter_type</i> ( <i>RID</i> voice ) const
<i>float</i>	<i>voice_get_filter_cutoff</i> ( <i>RID</i> voice ) const
<i>float</i>	<i>voice_get_filter_resonance</i> ( <i>RID</i> voice ) const
<i>float</i>	<i>voice_get_chorus</i> ( <i>RID</i> voice ) const
<i>int</i>	<i>voice_get_reverb_type</i> ( <i>RID</i> voice ) const
<i>float</i>	<i>voice_get_reverb</i> ( <i>RID</i> voice ) const
<i>int</i>	<i>voice_get_mix_rate</i> ( <i>RID</i> voice ) const
<i>bool</i>	<i>voice_is_positional</i> ( <i>RID</i> voice ) const
<i>void</i>	<i>voice_stop</i> ( <i>RID</i> voice )
<i>void</i>	<i>free_rid</i> ( <i>RID</i> rid )
<i>void</i>	<i>set_stream_global_volume_scale</i> ( <i>float</i> scale )

Continued on next page

Table 9.6 – continued from previous page

<code>float</code>	<code>get_stream_global_volume_scale()</code> const
<code>void</code>	<code>set_fx_global_volume_scale(float scale)</code>
<code>float</code>	<code>get_fx_global_volume_scale()</code> const
<code>void</code>	<code>set_event_voice_global_volume_scale(float scale)</code>
<code>float</code>	<code>get_event_voice_global_volume_scale()</code> const

### 9.14.3 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.

### 9.14.4 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.

### 9.14.5 Member Function Description

- `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.

- `void sample_set_description ( RID sample, String description )`

Set the description of an audio sample. Mainly used for organization.

- `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.

- `bool sample_is_stereo ( RID sample ) const`

Return whether the sample is stereo (2 channels).

- `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.

- `void sample_set_signed_data ( RID sample, RealArray data )`

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.

- `void sample_set_data ( RID sample, RawArray data )`

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:

Get the sample length (`get_sample_length`).

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.

- `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.

- `void sample_set_mix_rate ( RID sample, int mix_rate )`

Change the default mix rate of a given sample.

- `int sample_get_mix_rate ( RID sample ) const`

Return the mix rate of the given sample.

- `void sample_set_loop_format ( RID sample, int 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.

- `int sample_get_loop_format ( RID sample ) const`

Return the loop format for a sample, as a value from the SAMPLE\_LOOP\_\* enum.

- `void sample_set_loop_begin ( RID sample, int pos )`

Set the initial loop point of a sample. Only has effect if sample loop is enabled. See `sample_set_loop_format`.

- `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`.

- `void sample_set_loop_end ( RID sample, int pos )`

Set the final 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](#).

- `RID voice_create ()`

Allocate a voice for playback. Voices are persistent. A voice can play a single sample at the same time. See [sample\\_create](#).

- `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_volume ( RID voice, float volume )`

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_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_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_chorus ( RID voice, float chorus )`

Set chorus send post processing for the voice (from 0 to 1).

- `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.

- `void voice_set_mix_rate ( RID voice, int rate )`

Set a different playback mix rate for the given voice.

- `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.

- `float voice_get_volume ( RID voice ) const`

Return the current volume 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_height ( RID voice ) const`

Return the current pan height 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).

- `int voice_get_filter_type ( RID voice ) const`

Return the current selected filter type for a given voice, from the FILTER\_\* enum.

- `float voice_get_filter_cutoff ( RID voice ) const`

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.

- `float voice_get_chorus ( RID voice ) const`

Return the current chorus 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_reverb ( RID voice ) const`

Return the current reverb send for a given voice (0 to 1).

- `int voice_get_mix_rate ( RID voice ) const`

Return the current mix rate for a given voice.

- `bool voice_is_positional ( RID voice ) const`

Return whether the current voice is positional. See `voice_set_positional`.

- `void voice_stop ( RID voice )`

Stop a given voice.

- `void free_rid ( RID rid )`

Free a `RID` resource.

- `void set_stream_global_volume_scale ( float scale )`

Set global scale for stream playback. Default is 1.0.

- `float get_stream_global_volume_scale ( ) const`

Return the global scale for stream playback.

- `void set_fx_global_volume_scale ( float scale )`

Set global scale for all voices (not including streams). Default is 1.0.

- `float get_fx_global_volume_scale ( ) const`

Return the global scale for all voices.

- `void set_event_voice_global_volume_scale ( float scale )`

Set global scale for event-based stream (`EventStream`) playback. Default is 1.0.

- `float get_event_voice_global_volume_scale ( ) const`

Return the global scale for event-based stream playback.

## 9.15 AudioServerSW

Inherits: `AudioServer < Object`

Category: Core

### 9.15.1 Brief Description

Software implementation of `AudioServer`.

## 9.15.2 Description

This is a software audio server. It does not use any kind of hardware acceleration.

This class does not expose any new method.

# 9.16 AudioStream

**Inherits:** [Resource](#) < [Reference](#) < [Object](#)

**Inherited By:** [AudioStreamSpeex](#), [AudioStreamMPC](#), [AudioStreamOGGVorbis](#), [AudioStreamOpus](#)

**Category:** Core

## 9.16.1 Brief Description

Base class for audio streams.

## 9.16.2 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](#).

# 9.17 AudioStreamMPC

**Inherits:** [AudioStream](#) < [Resource](#) < [Reference](#) < [Object](#)

**Category:** Core

## 9.17.1 Brief Description

MusePack audio stream driver.

## 9.17.2 Description

MusePack audio stream driver.

# 9.18 AudioStreamOGGVorbis

**Inherits:** [AudioStream](#) < [Resource](#) < [Reference](#) < [Object](#)

**Category:** Core

## 9.18.1 Brief Description

OGG Vorbis audio stream driver.

### 9.18.2 Description

OGG Vorbis audio stream driver.

## 9.19 AudioStreamOpus

**Inherits:** *AudioStream < Resource < Reference < Object*

**Category:** Core

### 9.19.1 Brief Description

Opus Codec audio stream driver.

### 9.19.2 Description

Opus Codec audio stream driver.

## 9.20 AudioStreamPlayback

**Inherits:** *Reference < Object*

**Category:** Core

### 9.20.1 Brief Description

### 9.20.2 Member Functions

void	<i>play</i> ( <i>float</i> from_pos_sec=0)
void	<i>stop</i> ()
<i>bool</i>	<i>is_playing</i> () const
void	<i>set_loop</i> ( <i>bool</i> enabled)
<i>bool</i>	<i>has_loop</i> () const
<i>int</i>	<i>get_loop_count</i> () const
void	<i>seek_pos</i> ( <i>float</i> pos)
<i>float</i>	<i>get_pos</i> () const
<i>float</i>	<i>get_length</i> () const
<i>int</i>	<i>get_channels</i> () const
<i>int</i>	<i>get_mix_rate</i> () const
<i>int</i>	<i>get_minimum_buffer_size</i> () const

### 9.20.3 Member Function Description

- void **play** (*float* from\_pos\_sec=0)
- void **stop** ()
- *bool* **is\_playing** () const

- void `set_loop ( bool enabled )`
- `bool has_loop () const`
- `int get_loop_count () const`
- void `seek_pos ( float pos )`
- `float get_pos () const`
- `float get_length () const`
- `int get_channels () const`
- `int get_mix_rate () const`
- `int get_minimum_buffer_size () const`

## 9.21 AudioStreamSpeex

**Inherits:** `AudioStream < Resource < Reference < Object`

**Category:** Core

### 9.21.1 Brief Description

Speex audio stream driver.

### 9.21.2 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.

## 9.22 BackBufferCopy

**Inherits:** `Node2D < CanvasItem < Node < Object`

**Category:** Core

### 9.22.1 Brief Description

### 9.22.2 Member Functions

void	<code>set_rect ( Rect2 rect )</code>
<code>Rect2</code>	<code>get_rect () const</code>
void	<code>set_copy_mode ( int copy_mode )</code>
<code>int</code>	<code>get_copy_mode () const</code>

### 9.22.3 Numeric Constants

- COPY\_MODE\_DISABLED = 0
- COPY\_MODE\_RECT = 1
- COPY\_MODE\_VIEWPORT = 2

### 9.22.4 Member Function Description

- void `set_rect ( Rect2 rect )`
- `Rect2 get_rect () const`
- void `set_copy_mode ( int copy_mode )`
- `int get_copy_mode () const`

## 9.23 BakedLight

Inherits: *Resource < Reference < Object*

Category: Core

### 9.23.1 Brief Description

### 9.23.2 Member Functions

void	<code>set_mode ( int mode )</code>
<i>int</i>	<code>get_mode () const</code>
void	<code>set_octree ( RawArray octree )</code>
<i>RawArray</i>	<code>get_octree () const</code>
void	<code>set_light ( RawArray light )</code>
<i>RawArray</i>	<code>get_light () const</code>
void	<code>set_sampler_octree ( IntArray sampler_octree )</code>
<i>IntArray</i>	<code>get_sampler_octree () const</code>
void	<code>add_lightmap ( Texture texture, Vector2 gen_size )</code>
void	<code>erase_lightmap ( int id )</code>
void	<code>clear_lightmaps ()</code>
void	<code>set_cell_subdivision ( int cell_subdivision )</code>
<i>int</i>	<code>get_cell_subdivision () const</code>
void	<code>set_initial_lattice_subdiv ( int cell_subdivision )</code>
<i>int</i>	<code>get_initial_lattice_subdiv () const</code>
void	<code>set_plot_size ( float plot_size )</code>
<i>float</i>	<code>get_plot_size () const</code>
void	<code>set_bounces ( int bounces )</code>
<i>int</i>	<code>get_bounces () const</code>
void	<code>set_cell_extra_margin ( float cell_extra_margin )</code>
<i>float</i>	<code>get_cell_extra_margin () const</code>
void	<code>set_edge_damp ( float edge_damp )</code>
<i>float</i>	<code>get_edge_damp () const</code>

Continued on next page

Table 9.7 – continued from previous page

void	<code>set_normal_damp (float normal_damp)</code>
<i>float</i>	<code>get_normal_damp () const</code>
void	<code>set_tint (float tint)</code>
<i>float</i>	<code>get_tint () const</code>
void	<code>set_saturation (float saturation)</code>
<i>float</i>	<code>get_saturation () const</code>
void	<code>set_ao_radius (float ao_radius)</code>
<i>float</i>	<code>get_ao_radius () const</code>
void	<code>set_ao_strength (float ao_strength)</code>
<i>float</i>	<code>get_ao_strength () const</code>
void	<code>set_format (int format)</code>
<i>int</i>	<code>get_format () const</code>
void	<code>set_transfer_lightmaps_only_to_uv2 (bool enable)</code>
<i>bool</i>	<code>get_transfer_lightmaps_only_to_uv2 () const</code>
void	<code>set_energy_multiplier (float energy_multiplier)</code>
<i>float</i>	<code>get_energy_multiplier () const</code>
void	<code>set_gamma_adjust (float gamma_adjust)</code>
<i>float</i>	<code>get_gamma_adjust () const</code>
void	<code>set_bake_flag (int flag, bool enabled)</code>
<i>bool</i>	<code>get_bake_flag (int flag) const</code>

### 9.23.3 Numeric Constants

- **MODE\_OCTREE = 0**
- **MODE\_LIGHTMAPS = 1**
- **BAKE\_DIFFUSE = 0**
- **BAKE\_SPECULAR = 1**
- **BAKE\_TRANSLUCENT = 2**
- **BAKE\_CONSERVE\_ENERGY = 3**
- **BAKE\_MAX = 5**

### 9.23.4 Member Function Description

- void `set_mode (int mode)`
- *int* `get_mode () const`
- void `set_octree (RawArray octree)`
- *RawArray* `get_octree () const`
- void `set_light (RawArray light)`
- *RawArray* `get_light () const`
- void `set_sampler_octree (IntArray sampler_octree)`
- *IntArray* `get_sampler_octree () const`
- void `add_lightmap (Texture texture, Vector2 gen_size)`
- void `erase_lightmap (int id)`

- void **clear\_lightmaps** ()
- void **set\_cell\_subdivision** ( *int* cell\_subdivision )
- *int* **get\_cell\_subdivision** ( ) const
- void **set\_initial\_lattice\_subdiv** ( *int* cell\_subdivision )
- *int* **get\_initial\_lattice\_subdiv** ( ) const
- void **set\_plot\_size** ( *float* plot\_size )
- *float* **get\_plot\_size** ( ) const
- void **set\_bounces** ( *int* bounces )
- *int* **get\_bounces** ( ) const
- void **set\_cell\_extra\_margin** ( *float* cell\_extra\_margin )
- *float* **get\_cell\_extra\_margin** ( ) const
- void **set\_edge\_damp** ( *float* edge\_damp )
- *float* **get\_edge\_damp** ( ) const
- void **set\_normal\_damp** ( *float* normal\_damp )
- *float* **get\_normal\_damp** ( ) const
- void **set\_tint** ( *float* tint )
- *float* **get\_tint** ( ) const
- void **set\_saturation** ( *float* saturation )
- *float* **get\_saturation** ( ) const
- void **set\_ao\_radius** ( *float* ao\_radius )
- *float* **get\_ao\_radius** ( ) const
- void **set\_ao\_strength** ( *float* ao\_strength )
- *float* **get\_ao\_strength** ( ) const
- void **set\_format** ( *int* format )
- *int* **get\_format** ( ) const
- void **set\_transfer\_lightmaps\_only\_to\_uv2** ( *bool* enable )
- *bool* **get\_transfer\_lightmaps\_only\_to\_uv2** ( ) const
- void **set\_energy\_multiplier** ( *float* energy\_multiplier )
- *float* **get\_energy\_multiplier** ( ) const
- void **set\_gamma\_adjust** ( *float* gamma\_adjust )
- *float* **get\_gamma\_adjust** ( ) const
- void **set\_bake\_flag** ( *int* flag, *bool* enabled )
- *bool* **get\_bake\_flag** ( *int* flag ) const

## 9.24 BakedLightInstance

**Inherits:** *VisualInstance < Spatial < Node < Object*

**Category:** Core

### 9.24.1 Brief Description

### 9.24.2 Member Functions

void	<code>set_baked_light ( Object baked_light )</code>
<i>Object</i>	<code>get_baked_light () const</code>
<i>RID</i>	<code>get_baked_light_instance () const</code>

### 9.24.3 Signals

- `baked_light_changed ()`

### 9.24.4 Member Function Description

- void `set_baked_light ( Object baked_light )`
- *Object* `get_baked_light () const`
- *RID* `get_baked_light_instance () const`

## 9.25 BakedLightSampler

**Inherits:** *VisualInstance < Spatial < Node < Object*

**Category:** Core

### 9.25.1 Brief Description

### 9.25.2 Member Functions

void	<code>set_param ( int param, float value )</code>
<i>float</i>	<code>get_param ( int param ) const</code>
void	<code>set_resolution ( int resolution )</code>
<i>int</i>	<code>get_resolution () const</code>

### 9.25.3 Numeric Constants

- **PARAM\_RADIUS = 0**
- **PARAM\_STRENGTH = 1**
- **PARAM\_ATTENUATION = 2**
- **PARAM\_DETAIL\_RATIO = 3**

- **PARAM\_MAX = 4**

## 9.25.4 Member Function Description

- void **set\_param** ( *int* param, *float* value )
- *float* **get\_param** ( *int* param ) const
- void **set\_resolution** ( *int* resolution )
- *int* **get\_resolution** ( ) const

## 9.26 BaseButton

**Inherits:** *Control* < *CanvasItem* < *Node* < *Object*

**Inherited By:** *TextureButton*, *Button*

**Category:** Core

### 9.26.1 Brief Description

Provides a base class for different kinds of buttons.

### 9.26.2 Member Functions

void	<b>_pressed</b> ( ) virtual
void	<b>_toggled</b> ( <i>bool</i> pressed ) virtual
void	<b>set_pressed</b> ( <i>bool</i> pressed )
<i>bool</i>	<b>is_pressed</b> ( ) const
<i>bool</i>	<b>is_hovered</b> ( ) const
void	<b>set_toggle_mode</b> ( <i>bool</i> enabled )
<i>bool</i>	<b>is_toggle_mode</b> ( ) const
void	<b>set_disabled</b> ( <i>bool</i> disabled )
<i>bool</i>	<b>is_disabled</b> ( ) const
void	<b>set_click_on_press</b> ( <i>bool</i> enable )
<i>bool</i>	<b>get_click_on_press</b> ( ) const
<i>int</i>	<b>get_draw_mode</b> ( ) const

### 9.26.3 Signals

- **released** ( )
- **toggled** ( *bool* pressed )
- **pressed** ( )

## 9.26.4 Numeric Constants

- **DRAW\_NORMAL = 0**
- **DRAW\_PRESSED = 1**
- **DRAW\_HOVER = 2**
- **DRAW\_DISABLED = 3**

## 9.26.5 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.

## 9.26.6 Member Function Description

- void **\_pressed()** virtual
- void **\_toggled( bool pressed )** virtual
- void **set\_pressed( bool pressed )**

Set the button to pressed state (only if toggle\_mode is active).

- *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\_hovered()** const
- 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.

- *bool* **is\_toggle\_mode()** const

Return the toggle\_mode property (see [set\\_toggle\\_mode](#)).

- void **set\_disabled( bool disabled )**

Set the button into disabled state. When a button is disabled, it can't be clicked or toggled.

- *bool* **is\_disabled()** const

Return whether the button is in disabled state (see [set\\_disabled](#)).

- 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).

- *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.

## 9.27 BitMap

Inherits: [Resource](#) < [Reference](#) < [Object](#)

Category: Core

### 9.27.1 Brief Description

### 9.27.2 Member Functions

void	<code>create ( Vector2 size )</code>
void	<code>create_from_image_alpha ( Image image )</code>
void	<code>set_bit ( Vector2 pos, bool bit )</code>
<i>bool</i>	<code>get_bit ( Vector2 pos ) const</code>
void	<code>set_bit_rect ( Rect2 p_rect, bool bit )</code>
<i>int</i>	<code>get_true_bit_count () const</code>
<i>Vector2</i>	<code>get_size () const</code>

### 9.27.3 Member Function Description

- void `create ( Vector2 size )`
- void `create_from_image_alpha ( Image image )`
- void `set_bit ( Vector2 pos, bool bit )`
- *bool* `get_bit ( Vector2 pos ) const`
- void `set_bit_rect ( Rect2 p_rect, bool bit )`
- *int* `get_true_bit_count () const`
- *Vector2* `get_size () const`

## 9.28 BoneAttachment

Inherits: [Spatial](#) < [Node](#) < [Object](#)

Category: Core

### 9.28.1 Brief Description

## 9.29 bool

Category: Built-In Types

### 9.29.1 Brief Description

Boolean built-in type

## 9.29.2 Member Functions

<code>bool</code>	<code>bool ( int from )</code>
<code>bool</code>	<code>bool ( float from )</code>
<code>bool</code>	<code>bool ( String from )</code>

## 9.29.3 Description

Boolean built-in type.

## 9.29.4 Member Function Description

- `bool bool ( int from )`
- `bool bool ( float from )`
- `bool bool ( String from )`

## 9.30 BoxContainer

**Inherits:** `Container < Control < CanvasItem < Node < Object`

**Inherited By:** `VBoxContainer, ButtonGroup, HBoxContainer, ColorPicker`

**Category:** Core

### 9.30.1 Brief Description

Base class for Box containers.

### 9.30.2 Member Functions

<code>int</code>	<code>get_alignment () const</code>
<code>void</code>	<code>set_alignment ( int alignment )</code>

### 9.30.3 Numeric Constants

- `ALIGN_BEGIN = 0`
- `ALIGN_CENTER = 1`
- `ALIGN_END = 2`

### 9.30.4 Description

Base class for Box containers. It arranges children controls vertically or horizontally, and rearranges them automatically when their minimum size changes.

## 9.30.5 Member Function Description

- `int get_alignment() const`
- `void set_alignment( int alignment )`

## 9.31 BoxShape

**Inherits:** `Shape < Resource < Reference < Object`

**Category:** Core

### 9.31.1 Brief Description

Box shape resource.

### 9.31.2 Member Functions

<code>void</code>	<code>set_extents( Vector3 extents )</code>
<code>Vector3</code>	<code>get_extents() const</code>

### 9.31.3 Description

Box shape resource, which can be set into a `PhysicsBody` or area.

### 9.31.4 Member Function Description

- `void set_extents( Vector3 extents )`

Set the half extents for the shape.

- `Vector3 get_extents() const`

Return the half extents of the shape.

## 9.32 Button

**Inherits:** `BaseButton < Control < CanvasItem < Node < Object`

**Inherited By:** `OptionButton, ColorPickerButton, CheckButton, MenuButton, ToolButton, CheckBox`

**Category:** Core

### 9.32.1 Brief Description

Standard themed Button.

## 9.32.2 Member Functions

void	<code>set_text ( String text )</code>
<i>String</i>	<code>get_text () const</code>
void	<code>set_button_icon ( Texture texture )</code>
<i>Texture</i>	<code>get_button_icon () const</code>
void	<code>set_flat ( bool enabled )</code>
void	<code>set_clip_text ( bool enabled )</code>
<i>bool</i>	<code>get_clip_text () const</code>
void	<code>set_text_align ( int align )</code>
<i>int</i>	<code>get_text_align () const</code>
<i>bool</i>	<code>is_flat () const</code>

## 9.32.3 Description

Button is just the standard themed button: image src="images/button\_example.png"/ It can contain text and an icon, and will display them according to the current *Theme*.

## 9.32.4 Member Function Description

- void `set_text ( String text )`

Set the button text, which will be displayed inside the button area.

- *String* `get_text () const`

Return the button text.

- void `set_button_icon ( Texture texture )`
- *Texture* `get_button_icon () const`
- void `set_flat ( bool enabled )`

Set the *flat* property of a Button. Flat buttons don't display decoration unless hovered or pressed.

- 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.

- *bool* `get_clip_text () const`

Return the state of the *clip\_text* property (see `set_clip_text`)

- void `set_text_align ( int align )`
- *int* `get_text_align () const`
- *bool* `is_flat () const`

Return the state of the *flat* property (see `set_flat`)

## 9.33 ButtonArray

**Inherits:** *Control < CanvasItem < Node < Object*

**Inherited By:** *HButtonArray, VButtonArray*

**Category:** Core

### 9.33.1 Brief Description

Array of Buttons.

### 9.33.2 Member Functions

void	<code>add_button ( String text )</code>
void	<code>add_icon_button ( Object icon, String text="" )</code>
void	<code>set_button_text ( int button, String text )</code>
void	<code>set_button_icon ( int button, Object icon )</code>
<code>String</code>	<code>get_button_text ( int button ) const</code>
<code>Object</code>	<code>get_button_icon ( int button ) const</code>
<code>int</code>	<code>get_button_count () const</code>
<code>int</code>	<code>get_selected () const</code>
<code>int</code>	<code>get_hovered () const</code>
void	<code>set_selected ( int button )</code>
void	<code>erase_button ( int button )</code>
void	<code>clear ()</code>

### 9.33.3 Signals

- `button_selected ( int button )`

### 9.33.4 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.

### 9.33.5 Description

Array of Buttons. A Button array is useful to have an array of buttons laid out vertically or horizontally. Only one can be selected. This is useful for joy pad based interfaces and option menus.

### 9.33.6 Member Function Description

- void `add_button ( String text )`

Add a new button.

- void `add_icon_button ( Object icon, String text="" )`
- void `set_button_text ( int button, String text )`

- `void set_button_icon ( int button, Object icon )`

Set the icon of an existing button.

- `String get_button_text ( int button ) const`

Return the text of an existing button.

- `Object get_button_icon ( int button ) const`

Return the icon of an existing button.

- `int get_button_count ( ) const`

Return the amount of buttons in the array.

- `int get_selected ( ) const`

Return the currently selected button in the array.

- `int get_hovered ( ) const`

Return the currently hovered button in the array.

- `void set_selected ( int button )`

Select a button in the array.

- `void erase_button ( int button )`

Remove a button in the array, by index.

- `void clear ( )`

Clear the button array.

## 9.34 ButtonGroup

**Inherits:** `BoxContainer < Container < Control < CanvasItem < Node < Object`

**Category:** Core

### 9.34.1 Brief Description

Group of Buttons.

### 9.34.2 Member Functions

<code>BaseButton</code>	<code>get_pressed_button ( ) const</code>
<code>int</code>	<code>get_pressed_button_index ( ) const</code>
<code>BaseButton</code>	<code>get_focused_button ( ) const</code>
<code>Array</code>	<code>get_button_list ( ) const</code>
<code>void</code>	<code>set_pressed_button ( BaseButton button )</code>

### 9.34.3 Description

Group of `Button`. All direct and indirect children buttons become radios. Only one allows being pressed.

## 9.34.4 Member Function Description

- `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).

- `BaseButton get_focused_button () const`

Return the focused button.

- `Array get_button_list () const`

Return the list of all the buttons in the group.

- `void set_pressed_button ( BaseButton button )`

Set the button to be pressed.

## 9.35 Camera

**Inherits:** *Spatial < Node < Object*

**Inherited By:** *InterpolatedCamera*

**Category:** Core

### 9.35.1 Brief Description

Camera node, displays from a point of view.

## 9.35.2 Member Functions

<code>Vector3</code>	<code>project_ray_normal ( Vector2 screen_point ) const</code>
<code>Vector3</code>	<code>project_local_ray_normal ( Vector2 screen_point ) const</code>
<code>Vector3</code>	<code>project_ray_origin ( Vector2 screen_point ) const</code>
<code>Vector2</code>	<code>unproject_position ( Vector3 world_point ) const</code>
<code>bool</code>	<code>is_position_behind ( Vector3 world_point ) const</code>
<code>Vector3</code>	<code>project_position ( Vector2 screen_point ) const</code>
<code>void</code>	<code>set_perspective ( float fov, float z_near, float z_far )</code>
<code>void</code>	<code>set_orthogonal ( float size, float z_near, float z_far )</code>
<code>void</code>	<code>make_current ()</code>
<code>void</code>	<code>clear_current ()</code>
<code>bool</code>	<code>is_current () const</code>
<code>Transform</code>	<code>get_camera_transform () const</code>
<code>float</code>	<code>get_fov () const</code>
<code>float</code>	<code>get_size () const</code>
<code>float</code>	<code>get_zfar () const</code>
<code>float</code>	<code>get_znear () const</code>
<code>int</code>	<code>get_projection () const</code>
<code>void</code>	<code>set_visible_layers ( int mask )</code>
<code>int</code>	<code>get_visible_layers () const</code>
<code>void</code>	<code>set_environment ( Environment env )</code>
<code>Environment</code>	<code>get_environment () const</code>
<code>void</code>	<code>set_keep_aspect_mode ( int mode )</code>
<code>int</code>	<code>get_keep_aspect_mode () const</code>

## 9.35.3 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**

## 9.35.4 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.

## 9.35.5 Member Function Description

- `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_local_ray_normal ( Vector2 screen_point ) const`
- `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.

- `Vector2 unproject_position ( Vector3 world_point ) const`

Return how a 3D point in worldspace maps to a 2D coordinate in the `Viewport` rectangle.

- `bool is_position_behind ( Vector3 world_point ) const`
- `Vector3 project_position ( Vector2 screen_point ) const`
- `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_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 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.

- `void clear_current ()`
- `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).

- `Transform get_camera_transform () const`

Get the camera transform. Subclassed cameras (such as CharacterCamera) may provide different transforms than the `Node` transform.

- `float get_fov () const`
- `float get_size () const`
- `float get_zfar () const`
- `float get_znear () const`
- `int get_projection () const`
- `void set_visible_layers ( int mask )`
- `int get_visible_layers () const`
- `void set_environment ( Environment env )`
- `Environment get_environment () const`
- `void set_keep_aspect_mode ( int mode )`
- `int get_keep_aspect_mode () const`

## 9.36 Camera2D

**Inherits:** [Node2D](#) < [CanvasItem](#) < [Node](#) < [Object](#)

**Category:** Core

### 9.36.1 Brief Description

Camera node for 2D scenes.

### 9.36.2 Member Functions

void	<code>set_offset ( Vector2 offset )</code>
<code>Vector2</code>	<code>get_offset () const</code>
void	<code>set_anchor_mode ( int anchor_mode )</code>
<code>int</code>	<code>get_anchor_mode () const</code>
void	<code>set_rotating ( bool rotating )</code>
<code>bool</code>	<code>is_rotating () const</code>
void	<code>make_current ()</code>
void	<code>clear_current ()</code>
<code>bool</code>	<code>is_current () const</code>
void	<code>set_limit ( int margin, int limit )</code>
<code>int</code>	<code>get_limit ( int margin ) const</code>
void	<code>set_v_drag_enabled ( bool enabled )</code>
<code>bool</code>	<code>is_v_drag_enabled () const</code>
void	<code>set_h_drag_enabled ( bool enabled )</code>
<code>bool</code>	<code>is_h_drag_enabled () const</code>
void	<code>set_v_offset ( float ofs )</code>
<code>float</code>	<code>get_v_offset () const</code>
void	<code>set_h_offset ( float ofs )</code>
<code>float</code>	<code>get_h_offset () const</code>
void	<code>set_drag_margin ( int margin, float drag_margin )</code>
<code>float</code>	<code>get_drag_margin ( int margin ) const</code>
<code>Vector2</code>	<code>get_camera_pos () const</code>
<code>Vector2</code>	<code>get_camera_screen_center () const</code>
void	<code>set_zoom ( Vector2 zoom )</code>
<code>Vector2</code>	<code>get_zoom () const</code>
void	<code>set_follow_smoothing ( float follow_smoothing )</code>
<code>float</code>	<code>get_follow_smoothing () const</code>
void	<code>set_enable_follow_smoothing ( bool follow_smoothing )</code>
<code>bool</code>	<code>is_follow_smoothing_enabled () const</code>
void	<code>force_update_scroll ()</code>

### 9.36.3 Numeric Constants

- `ANCHOR_MODE_DRAG_CENTER = 1`
- `ANCHOR_MODE_FIXED_TOP_LEFT = 0`

## 9.36.4 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](#).

## 9.36.5 Member Function Description

- `void set_offset ( Vector2 offset )`

Set the scroll offset. Useful for looking around or camera shake animations.

- `Vector2 get_offset () const`

Return the scroll offset.

- `void set_anchor_mode ( int anchor_mode )`
- `int get_anchor_mode () const`
- `void set_rotating ( bool rotating )`
- `bool is_rotating () 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 clear_current ()`
- `bool is_current () const`

Return true if this is the current camera (see [make\\_current](#)).

- `void set_limit ( int margin, int limit )`

Set the scrolling limit in pixels.

- `int get_limit ( int margin ) const`

Return the scrolling limit in pixels.

- `void set_v_drag_enabled ( bool enabled )`
- `bool is_v_drag_enabled () const`
- `void set_h_drag_enabled ( bool enabled )`
- `bool is_h_drag_enabled () const`
- `void set_v_offset ( float ofs )`
- `float get_v_offset () const`
- `void set_h_offset ( float ofs )`
- `float get_h_offset () const`
- `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.

- `float get_drag_margin ( int margin ) const`

Return the margins needed to drag the camera (see `set_drag_margin`).

- `Vector2 get_camera_pos () const`

Return the camera position.

- `Vector2 get_camera_screen_center () const`
- `void set_zoom ( Vector2 zoom )`
- `Vector2 get_zoom () const`
- `void set_follow_smoothing ( float follow_smoothing )`
- `float get_follow_smoothing () const`
- `void set_enable_follow_smoothing ( bool follow_smoothing )`
- `bool is_follow_smoothing_enabled () const`
- `void force_update_scroll ()`

Force the camera to update scroll immediately.

## 9.37 CanvasItem

**Inherits:** `Node < Object`

**Inherited By:** `Node2D, Control`

**Category:** Core

### 9.37.1 Brief Description

Base class of anything 2D.

### 9.37.2 Member Functions

<code>void</code>	<code>_draw () virtual</code>
<code>void</code>	<code>edit_set_state ( var state )</code>
<code>void</code>	<code>edit_get () const</code>
<code>void</code>	<code>edit_set_rect ( Rect2 rect )</code>
<code>void</code>	<code>edit_rotate ( float degrees )</code>
<code>Rect2</code>	<code>get_item_rect () const</code>
<code>RID</code>	<code>get_canvas_item () const</code>
<code>bool</code>	<code>is_visible () const</code>
<code>bool</code>	<code>is_hidden () const</code>
<code>void</code>	<code>show ()</code>
<code>void</code>	<code>hide ()</code>
<code>void</code>	<code>set_hidden ( bool hidden )</code>
<code>void</code>	<code>update ()</code>
<code>void</code>	<code>set_as_toplevel ( bool enable )</code>
<code>bool</code>	<code>is_set_as_toplevel () const</code>

Continue

Table 9.8 – continued from previous page

void	<code>set_blend_mode ( int blend_mode )</code>
<i>int</i>	<code>get_blend_mode () const</code>
void	<code>set_light_mask ( int light_mask )</code>
<i>int</i>	<code>get_light_mask () const</code>
void	<code>set_opacity ( float opacity )</code>
<i>float</i>	<code>get_opacity () const</code>
void	<code>set_self_opacity ( float self_opacity )</code>
<i>float</i>	<code>get_self_opacity () const</code>
void	<code>set_draw_behind_parent ( bool enable )</code>
<i>bool</i>	<code>is_draw_behind_parent_enabled () const</code>
void	<code>draw_line ( Vector2 from, Vector2 to, Color color, float width=1 )</code>
void	<code>draw_rect ( Rect2 rect, Color color )</code>
void	<code>draw_circle ( Vector2 pos, float radius, Color color )</code>
void	<code>draw_texture ( Texture texture, Vector2 pos, Color modulate=Color(1,1,1,1) )</code>
void	<code>draw_texture_rect ( Texture texture, Rect2 rect, bool tile, Color modulate=Color(1,1,1,1), bool transpose=false )</code>
void	<code>draw_texture_rect_region ( Texture texture, Rect2 rect, Rect2 src_rect, Color modulate=Color(1,1,1,1), bool transpose=false )</code>
void	<code>draw_style_box ( StyleBox style_box, Rect2 rect )</code>
void	<code>draw_primitive ( Vector2Array points, ColorArray colors, Vector2Array uvs=Array(), Texture texture=Object() )</code>
void	<code>draw_polygon ( Vector2Array points, ColorArray colors, Vector2Array uvs=Array(), Texture texture=Object() )</code>
void	<code>draw_colored_polygon ( Vector2Array points, Color color, Vector2Array uvs=Array(), Texture texture=Object() )</code>
void	<code>draw_string ( Font font, Vector2 pos, String text, Color modulate=Color(1,1,1,1), int clip_w=-1 )</code>
<i>float</i>	<code>draw_char ( Font font, Vector2 pos, String char, String next, Color modulate=Color(1,1,1,1) )</code>
void	<code>draw_set_transform ( Vector2 pos, float rot, Vector2 scale )</code>
<i>Matrix32</i>	<code>get_transform () const</code>
<i>Matrix32</i>	<code>get_global_transform () const</code>
<i>Matrix32</i>	<code>get_global_transform_with_canvas () const</code>
<i>Matrix32</i>	<code>get_viewport_transform () const</code>
<i>Rect2</i>	<code>get_viewport_rect () const</code>
<i>Matrix32</i>	<code>get_canvas_transform () const</code>
<i>Vector2</i>	<code>get_local_mouse_pos () const</code>
<i>Vector2</i>	<code>get_global_mouse_pos () const</code>
<i>RID</i>	<code>get_canvas () const</code>
<i>Object</i>	<code>get_world_2d () const</code>
void	<code>set_material ( CanvasItemMaterial material )</code>
<i>CanvasItemMaterial</i>	<code>get_material () const</code>
void	<code>set_use_parent_material ( bool enable )</code>
<i>bool</i>	<code>get_use_parent_material () const</code>
<i>InputEvent</i>	<code>make_input_local ( InputEvent event ) const</code>

### 9.37.3 Signals

- `item_rect_changed ()`
- `draw ()`
- `visibility_changed ()`
- `hide ()`

## 9.37.4 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.

## 9.37.5 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 significan’tly. 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.

## 9.37.6 Member Function Description

- void **\_draw ()** virtual

Called (if exists) to draw the canvas item.

- void **edit\_set\_state ( var state )**

Used for editing, returns an opaque value representing the transform state.

- void **edit\_get () const**
- void **edit\_set\_rect ( Rect2 rect )**
- void **edit\_rotate ( float degrees )**

Used for editing, handle rotation.

- **Rect2 get\_item\_rect () const**

Return a rect containing the editable contents of the item.

- `RID get_canvas_item () const`

Return the canvas item RID used by `VisualServer` for this item.

- `bool is_visible () const`

Return true if this CanvasItem is visible. It may be invisible because itself or a parent canvas item is hidden.

- `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.

- `void show ()`

Show the CanvasItem currently hidden.

- `void hide ()`

Hide the CanvasItem currently visible.

- `void set_hidden ( bool hidden )`

- `void update ()`

Queue the CanvasItem for update. `NOTIFICATION_DRAW` will be called on idle time to request redraw.

- `void set_as_toplevel ( bool enable )`

Set as toplevel. This means that it will not inherit transform from parent canvas items.

- `bool is_set_as_toplevel () const`

Return if set as toplevel. See `set_as_toplevel`.

- `void set_blend_mode ( int blend_mode )`

Set the blending mode from enum `BLEND_MODE_*`.

- `int get_blend_mode () const`

Return the current blending mode from enum `BLEND_MODE_*`.

- `void set_light_mask ( int light_mask )`

- `int get_light_mask () const`

- `void set_opacity ( float opacity )`

Set canvas item opacity. This will affect the canvas item and all the children.

- `float get_opacity () const`

Return the canvas item opacity. This affects 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.

- `float get_self_opacity () const`

Return the canvas item self-opacity.

- `void set_draw_behind_parent ( bool enable )`

Sets whether the canvas item is drawn behind its parent.

- `bool is_draw_behind_parent_enabled () const`

Return whether the item is drawn behind its parent.

- `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_rect ( Rect2 rect, Color color )`

Draw a colored rectangle.

- `void draw_circle ( Vector2 pos, float radius, Color color )`

Draw a colored circle.

- `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, bool tile, 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.

- `void draw_style_box ( StyleBox style_box, Rect2 rect )`

Draw a styled rectangle.

- `void draw_primitive ( Vector2Array points, ColorArray colors, Vector2Array uvs=Array(), Texture texture=Object(), 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_polygon ( Vector2Array points, ColorArray colors, Vector2Array uvs=Array(), Texture texture=Object() )`

Draw a polygon of any amount of points, convex or concave.

- `void draw_colored_polygon ( Vector2Array points, Color color, Vector2Array uvs=Array(), Texture texture=Object() )`

Draw a colored polygon of any amount of points, convex or concave.

- `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.

- `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_set_transform ( Vector2 pos, float rot, Vector2 scale )`

Set a custom transform for drawing. Anything drawn afterwards will be transformed by this.

- `Matrix32 get_transform () const`
- `Matrix32 get_global_transform () const`
- `Matrix32 get_global_transform_with_canvas () const`
- `Matrix32 get_viewport_transform () const`

- `Rect2 get_viewport_rect () const`
- `Matrix32 get_canvas_transform () const`
- `Vector2 get_local_mouse_pos () const`
- `Vector2 get_global_mouse_pos () const`
- `RID get_canvas () const`
- `Object get_world_2d () const`
- `void set_material ( CanvasItemMaterial material )`
- `CanvasItemMaterial get_material () const`
- `void set_use_parent_material ( bool enable )`
- `bool get_use_parent_material () const`
- `InputEvent make_input_local ( InputEvent event ) const`

## 9.38 CanvasItemMaterial

Inherits: `Resource < Reference < Object`

Category: Core

### 9.38.1 Brief Description

### 9.38.2 Member Functions

void	<code>set_shader ( Shader shader )</code>
<code>Shader</code>	<code>get_shader () const</code>
void	<code>set_shader_param ( String param, var value )</code>
void	<code>get_shader_param ( String param ) const</code>
void	<code>set_shading_mode ( int mode )</code>
<code>int</code>	<code>get_shading_mode () const</code>

### 9.38.3 Numeric Constants

- `SHADING_NORMAL = 0`
- `SHADING_UNSHADED = 1`
- `SHADING_ONLY_LIGHT = 2`

### 9.38.4 Member Function Description

- `void set_shader ( Shader shader )`
- `Shader get_shader () const`
- `void set_shader_param ( String param, var value )`
- `void get_shader_param ( String param ) const`
- `void set_shading_mode ( int mode )`

- `int get_shading_mode() const`

## 9.39 CanvasItemShader

**Inherits:** [Shader](#) < [Resource](#) < [Reference](#) < [Object](#)

**Category:** Core

### 9.39.1 Brief Description

## 9.40 CanvasItemShaderGraph

**Inherits:** [ShaderGraph](#) < [Shader](#) < [Resource](#) < [Reference](#) < [Object](#)

**Category:** Core

### 9.40.1 Brief Description

## 9.41 CanvasLayer

**Inherits:** [Node](#) < [Object](#)

**Inherited By:** [ParallaxBackground](#)

**Category:** Core

### 9.41.1 Brief Description

Canvas Item layer.

### 9.41.2 Member Functions

<code>void</code>	<code>set_layer( int layer )</code>
<code>int</code>	<code>get_layer() const</code>
<code>void</code>	<code>set_transform( <a href="#">Matrix32</a> transform )</code>
<code><a href="#">Matrix32</a></code>	<code>get_transform() const</code>
<code>void</code>	<code>set_offset( <a href="#">Vector2</a> offset )</code>
<code><a href="#">Vector2</a></code>	<code>get_offset() const</code>
<code>void</code>	<code>set_rotation( float rotation )</code>
<code>float</code>	<code>get_rotation() const</code>
<code>void</code>	<code>set_scale( <a href="#">Vector2</a> scale )</code>
<code><a href="#">Vector2</a></code>	<code>get_scale() const</code>
<code>Canvas</code>	<code>get_world_2d() const</code>
<code>RID</code>	<code>get_viewport() const</code>

### 9.41.3 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).

### 9.41.4 Member Function Description

- void [set\\_layer](#) ( [int](#) layer )

Set the layer index, determines the draw order, a lower value will be below a higher one.

- [int](#) [get\\_layer](#) ( ) const

Return the layer index, determines the draw order, a lower value will be below a higher one.

- void [set\\_transform](#) ( [Matrix32](#) transform )

Set the base transform for this layer.

- [Matrix32](#) [get\\_transform](#) ( ) const

Return the base transform for this layer.

- void [set\\_offset](#) ( [Vector2](#) offset )

Set the base offset for this layer (helper).

- [Vector2](#) [get\\_offset](#) ( ) const

Return the base offset for this layer (helper).

- void [set\\_rotation](#) ( [float](#) rotation )

Set the base rotation for this layer (helper).

- [float](#) [get\\_rotation](#) ( ) const

Return the base rotation for this layer (helper).

- void [set\\_scale](#) ( [Vector2](#) scale )

Set the base scale for this layer (helper).

- [Vector2](#) [get\\_scale](#) ( ) const

Return the base scale for this layer (helper).

- [Canvas](#) [get\\_world\\_2d](#) ( ) const

Return the [World2D](#) used by this layer.

- [RID](#) [get\\_viewport](#) ( ) const

Return the viewport RID for this layer.

## 9.42 CanvasModulate

**Inherits:** [Node2D](#) < [CanvasItem](#) < [Node](#) < [Object](#)

**Category:** Core

### 9.42.1 Brief Description

### 9.42.2 Member Functions

void	<code>set_color ( Color color )</code>
<code>Color</code>	<code>get_color () const</code>

### 9.42.3 Member Function Description

- void `set_color ( Color color )`
- `Color get_color () const`

## 9.43 CapsuleShape

**Inherits:** `Shape < Resource < Reference < Object`

**Category:** Core

### 9.43.1 Brief Description

Capsule shape resource.

### 9.43.2 Member Functions

void	<code>set_radius ( float radius )</code>
<code>float</code>	<code>get_radius () const</code>
void	<code>set_height ( float height )</code>
<code>float</code>	<code>get_height () const</code>

### 9.43.3 Description

Capsule shape resource, which can be set into a `PhysicsBody` or area.

### 9.43.4 Member Function Description

- void `set_radius ( float radius )`

Set the capsule radius.

- `float get_radius () const`

Return the capsule radius.

- void `set_height ( float height )`

Set the capsule height.

- `float get_height () const`

Return the capsule height.

## 9.44 CapsuleShape2D

Inherits: [Shape2D](#) < [Resource](#) < [Reference](#) < [Object](#)

Category: Core

### 9.44.1 Brief Description

Capsule 2D shape resource for physics.

### 9.44.2 Member Functions

void	<a href="#"><code>set_radius (float radius)</code></a>
<code>float</code>	<a href="#"><code>get_radius () const</code></a>
void	<a href="#"><code>set_height (float height)</code></a>
<code>float</code>	<a href="#"><code>get_height () const</code></a>

### 9.44.3 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.

### 9.44.4 Member Function Description

- void [`set\_radius \(float radius\)`](#)

Set the radius of the [CapsuleShape2D](#).

- `float` [`get\_radius \(\) const`](#)

Return the radius of the [CapsuleShape2D](#).

- void [`set\_height \(float height\)`](#)

Set the height of the [CapsuleShape2D](#).

- `float` [`get\_height \(\) const`](#)

Return the height of the [CapsuleShape2D](#).

## 9.45 CenterContainer

Inherits: [Container](#) < [Control](#) < [CanvasItem](#) < [Node](#) < [Object](#)

Category: Core

### 9.45.1 Brief Description

Keeps children controls centered.

## 9.45.2 Member Functions

void	<code>set_use_top_left ( bool enable )</code>
<code>bool</code>	<code>is_using_top_left () const</code>

## 9.45.3 Description

CenterContainer Keeps children controls centered. This container keeps all children to their minimum size, in the center.

## 9.45.4 Member Function Description

- void `set_use_top_left ( bool enable )`
- `bool is_using_top_left () const`

## 9.46 CheckBox

**Inherits:** `Button < BaseButton < Control < CanvasItem < Node < Object`

**Category:** Core

### 9.46.1 Brief Description

## 9.47 CheckButton

**Inherits:** `Button < BaseButton < Control < CanvasItem < Node < Object`

**Category:** Core

### 9.47.1 Brief Description

Checkable button.

### 9.47.2 Description

CheckButton is a toggle button displayed as a check field.

## 9.48 CircleShape2D

**Inherits:** `Shape2D < Resource < Reference < Object`

**Category:** Core

### 9.48.1 Brief Description

Circular Shape for 2D Physics.

### 9.48.2 Member Functions

void	<code>set_radius (float radius)</code>
float	<code>get_radius () const</code>

### 9.48.3 Description

Circular Shape for 2D Physics. This shape is useful for modeling balls or small characters and it's collision detection with everything else is very fast.

### 9.48.4 Member Function Description

- void `set_radius (float radius)`

Set the radius of the circle shape.

- float `get_radius () const`

Return the radius of the circle shape.

## 9.49 CollisionObject

**Inherits:** *Spatial < Node < Object*

**Inherited By:** *PhysicsBody, Area*

**Category:** Core

## 9.49.1 Brief Description

## 9.49.2 Member Functions

void	<code>_input_event ( Object camera, InputEvent event, Vector3 click_pos, Vector3 click_normal, int shape_idx ) virtual</code>
void	<code>add_shape ( Shape shape, Transform transform=Transform() )</code>
<code>int</code>	<code>get_shape_count () const</code>
void	<code>set_shape ( int shape_idx, Shape shape )</code>
void	<code>set_shape_transform ( int shape_idx, Transform transform )</code>
void	<code>set_shape_as_trigger ( int shape_idx, bool enable )</code>
<code>bool</code>	<code>is_shape_set_as_trigger ( int shape_idx ) const</code>
<code>Shape</code>	<code>get_shape ( int shape_idx ) const</code>
<code>Transform</code>	<code>get_shape_transform ( int shape_idx ) const</code>
void	<code>remove_shape ( int shape_idx )</code>
void	<code>clear_shapes ()</code>
void	<code>set_ray_pickable ( bool ray_pickable )</code>
<code>bool</code>	<code>is_ray_pickable () const</code>
void	<code>set_capture_input_on_drag ( bool enable )</code>
<code>bool</code>	<code>get_capture_input_on_drag () const</code>
<code>RID</code>	<code>get_rid () const</code>

## 9.49.3 Signals

- `mouse_enter ()`
- `input_event ( Object camera, InputEvent event, Vector3 click_pos, Vector3 click_normal, int shape_idx )`
- `mouse_exit ()`

## 9.49.4 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() )`
- `int get_shape_count () const`
- void `set_shape ( int shape_idx, Shape shape )`
- void `set_shape_transform ( int shape_idx, Transform transform )`
- void `set_shape_as_trigger ( int shape_idx, bool enable )`
- `bool is_shape_set_as_trigger ( int shape_idx ) const`
- `Shape get_shape ( int shape_idx ) const`
- `Transform get_shape_transform ( int shape_idx ) const`
- void `remove_shape ( int shape_idx )`
- void `clear_shapes ()`
- void `set_ray_pickable ( bool ray_pickable )`

- `bool is_ray_pickable () const`
- `void set_capture_input_on_drag ( bool enable )`
- `bool get_capture_input_on_drag () const`
- `RID get_rid () const`

## 9.50 CollisionObject2D

**Inherits:** `Node2D < CanvasItem < Node < Object`

**Inherited By:** `Area2D, PhysicsBody2D`

**Category:** Core

### 9.50.1 Brief Description

Base node for 2D collisionables.

### 9.50.2 Member Functions

<code>void</code>	<code>_input_event ( Object viewport, InputEvent event, int shape_idx ) virtual</code>
<code>void</code>	<code>add_shape ( Shape2D shape, Matrix32 transform=1,0, 0,1, 0,0 )</code>
<code>int</code>	<code>get_shape_count () const</code>
<code>void</code>	<code>set_shape ( int shape_idx, Shape shape )</code>
<code>void</code>	<code>set_shape_transform ( int shape_idx, Matrix32 transform )</code>
<code>void</code>	<code>set_shape_as_trigger ( int shape_idx, bool enable )</code>
<code>Shape2D</code>	<code>get_shape ( int shape_idx ) const</code>
<code>Matrix32</code>	<code>get_shape_transform ( int shape_idx ) const</code>
<code>bool</code>	<code>is_shape_set_as_trigger ( int shape_idx ) const</code>
<code>void</code>	<code>remove_shape ( int shape_idx )</code>
<code>void</code>	<code>clear_shapes ()</code>
<code>RID</code>	<code>get_rid () const</code>
<code>void</code>	<code>set_pickable ( bool enabled )</code>
<code>bool</code>	<code>is_pickable () const</code>

### 9.50.3 Signals

- `mouse_enter ()`
- `input_event ( Object viewport, InputEvent event, int shape_idx )`
- `mouse_exit ()`

### 9.50.4 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.

## 9.50.5 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.

- *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*.

- void `set_shape` ( *int* shape\_idx, *Shape* shape )

Change a shape in the collision body.

- void `set_shape_transform` ( *int* shape\_idx, *Matrix32* transform )

Change the shape transform 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).

- *Shape2D* `get_shape` ( *int* shape\_idx ) const

Return the shape in the given index.

- *Matrix32* `get_shape_transform` ( *int* shape\_idx ) const

Return the shape transform in the given index.

- *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 `clear_shapes` ( )

Remove all shapes.

- *RID* `get_rid` ( ) const

Return the RID of this object.

- 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.

- *bool* `is_pickable` ( ) const

Return whether this object is pickable.

## 9.51 CollisionPolygon

Inherits: [Spatial](#) < [Node](#) < [Object](#)

Category: Core

### 9.51.1 Brief Description

### 9.51.2 Member Functions

void	<code>set_build_mode ( int build_mode )</code>
<i>int</i>	<code>get_build_mode () const</code>
void	<code>set_depth ( float depth )</code>
<i>float</i>	<code>get_depth () const</code>
void	<code>set_polygon ( Vector2Array polygon )</code>
<i>Vector2Array</i>	<code>get_polygon () const</code>
<i>int</i>	<code>get_collision_object_first_shape () const</code>
<i>int</i>	<code>get_collision_object_last_shape () const</code>

### 9.51.3 Member Function Description

- void **set\_build\_mode** (*int* build\_mode)
- *int* **get\_build\_mode** () const
- void **set\_depth** (*float* depth)
- *float* **get\_depth** () const
- void **set\_polygon** (*Vector2Array* polygon)
- *Vector2Array* **get\_polygon** () const
- *int* **get\_collision\_object\_first\_shape** () const
- *int* **get\_collision\_object\_last\_shape** () const

## 9.52 CollisionPolygon2D

Inherits: [Node2D](#) < [CanvasItem](#) < [Node](#) < [Object](#)

Category: Core

### 9.52.1 Brief Description

Editor-only class for easy editing of collision polygons.

## 9.52.2 Member Functions

void	<code>set_polygon ( Vector2Array polygon )</code>
<code>Vector2Array</code>	<code>get_polygon () const</code>
void	<code>set_build_mode ( int build_mode )</code>
<code>int</code>	<code>get_build_mode () const</code>
void	<code>set_trigger ( bool trigger )</code>
<code>bool</code>	<code>is_trigger () const</code>
<code>int</code>	<code>get_collision_object_first_shape () const</code>
<code>int</code>	<code>get_collision_object_last_shape () const</code>

## 9.52.3 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.

## 9.52.4 Member Function Description

- 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.

- `Vector2Array get_polygon () const`

Return the list of points that define the polygon.

- void `set_build_mode ( int build_mode )`

Set whether the polygon is to be a `ConvexPolygon2D` (`build_mode==0`), or a `ConcavePolygon2D` (`build_mode==1`).

- `int get_build_mode () const`

Return whether the polygon is a `ConvexPolygon2D` (`build_mode==0`), or a `ConcavePolygon2D` (`build_mode==1`).

- void `set_trigger ( bool trigger )`

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).

- `bool is_trigger () const`

Return whether this polygon is a trigger.

- `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.

## 9.53 CollisionShape

**Inherits:** *Spatial < Node < Object*

**Category:** Core

### 9.53.1 Brief Description

### 9.53.2 Member Functions

void	<code>resource_changed ( Object resource )</code>
void	<code>set_shape ( Object shape )</code>
<i>Object</i>	<code>get_shape () const</code>
void	<code>set_trigger ( bool enable )</code>
<i>bool</i>	<code>is_trigger () const</code>
void	<code>make_convex_from_brothers ()</code>
<i>int</i>	<code>get_collision_object_shape_index () const</code>

### 9.53.3 Member Function Description

- void `resource_changed ( Object resource )`
- void `set_shape ( Object shape )`
- *Object* `get_shape () const`
- void `set_trigger ( bool enable )`
- *bool* `is_trigger () const`
- void `make_convex_from_brothers ()`
- *int* `get_collision_object_shape_index () const`

## 9.54 CollisionShape2D

**Inherits:** *Node2D < CanvasItem < Node < Object*

**Category:** Core

### 9.54.1 Brief Description

Editor-only class for easy editing of shapes.

## 9.54.2 Member Functions

void	<code>set_shape ( Object shape )</code>
<i>Object</i>	<code>get_shape () const</code>
void	<code>set_trigger ( bool enable )</code>
<i>bool</i>	<code>is_trigger () const</code>
<i>int</i>	<code>get_collision_object_shape_index () const</code>

## 9.54.3 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.

## 9.54.4 Member Function Description

- 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.

- *Object* `get_shape () const`

Return this shape's [Shape2D](#).

- 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).

- *bool* `is_trigger () const`

Return whether this shape is a trigger.

- *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](#).

## 9.55 Color

**Category:** Built-In Types

### 9.55.1 Brief Description

Color in RGBA format.

## 9.55.2 Member Functions

<i>Color</i>	<code>blend ( Color over )</code>
<i>Color</i>	<code>contrasted ()</code>
<i>float</i>	<code>gray ()</code>
<i>Color</i>	<code>inverted ()</code>
<i>Color</i>	<code>linear_interpolate ( Color b, float t )</code>
<i>int</i>	<code>to_32 ()</code>
<i>int</i>	<code>to_ARGB32 ()</code>
<i>String</i>	<code>to_html ( bool with_alpha=True )</code>
<i>Color</i>	<code>Color ( float r, float g, float b, float a )</code>
<i>Color</i>	<code>Color ( float r, float g, float b )</code>
<i>Color</i>	<code>Color ( int from )</code>
<i>Color</i>	<code>Color ( String from )</code>

## 9.55.3 Member Variables

- *float r*
- *float g*
- *float b*
- *float a*
- *float h*
- *float s*
- *float v*
- *int r8*
- *int g8*
- *int b8*
- *int a8*

## 9.55.4 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.

## 9.55.5 Member Function Description

- *Color blend ( Color over )*
- *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 its 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.

- `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.

## 9.56 ColorArray

**Category:** Built-In Types

### 9.56.1 Brief Description

Array of Colors

### 9.56.2 Member Functions

<code>void</code>	<code>push_back ( Color color )</code>
<code>void</code>	<code>resize ( int idx )</code>
<code>void</code>	<code>set ( int idx, Color color )</code>
<code>int</code>	<code>size ()</code>
<code>ColorArray</code>	<code>ColorArray ( Array from )</code>

### 9.56.3 Description

Array of Color, can only contains colors. Optimized for memory usage, can't fragment the memory.

## 9.56.4 Member Function Description

- void **push\_back** ( *Color* color )

Append a value to the array.

- void **resize** ( *int* idx )

Resize the array.

- void **set** ( *int* idx, *Color* color )

Set an index in the array.

- *int* **size** ()

Return the array size.

- *ColorArray* **ColorArray** ( *Array* from )

Create from a generic array.

## 9.57 ColorPicker

**Inherits:** *BoxContainer* < *Container* < *Control* < *CanvasItem* < *Node* < *Object*

**Category:** Core

### 9.57.1 Brief Description

Color picker control.

### 9.57.2 Member Functions

void	<i>set_color</i> ( <i>Color</i> color )
<i>Color</i>	<i>get_color</i> () const
void	<i>set_raw_mode</i> ( <i>bool</i> mode )
<i>bool</i>	<i>is_raw_mode</i> () const
void	<i>set_edit_alpha</i> ( <i>bool</i> show )
<i>bool</i>	<i>is_editing_alpha</i> () const
void	<i>add_preset</i> ( <i>Color</i> arg0 )

### 9.57.3 Signals

- **color\_changed** ( *Color* color )

### 9.57.4 Description

This is a simple color picker *Control*. It's useful for selecting a color from an RGB/RGBA colorspace.

## 9.57.5 Member Function Description

- void `set_color ( Color color )`

Select the current color.

- `Color get_color () const`

Return the current (edited) color.

- void `set_raw_mode ( bool mode )`
- `bool is_raw_mode () const`
- void `set_edit_alpha ( bool show )`
- `bool is_editing_alpha () const`
- void `add_preset ( Color arg0 )`

## 9.58 ColorPickerButton

**Inherits:** `Button < BaseButton < Control < CanvasItem < Node < Object`

**Category:** Core

### 9.58.1 Brief Description

### 9.58.2 Member Functions

void	<code>set_color ( Color color )</code>
<code>Color</code>	<code>get_color () const</code>
void	<code>set_edit_alpha ( bool show )</code>
<code>bool</code>	<code>is_editing_alpha () const</code>

### 9.58.3 Signals

- `color_changed ( Color color )`

### 9.58.4 Member Function Description

- void `set_color ( Color color )`
- `Color get_color () const`
- void `set_edit_alpha ( bool show )`
- `bool is_editing_alpha () const`

## 9.59 ColorRamp

**Inherits:** `Resource < Reference < Object`

**Category:** Core

### 9.59.1 Brief Description

### 9.59.2 Member Functions

void	<i>add_point</i> ( <i>float</i> offset, <i>Color</i> color )
void	<i>remove_point</i> ( <i>int</i> offset )
void	<i>set_offset</i> ( <i>int</i> point, <i>float</i> offset )
<i>float</i>	<i>get_offset</i> ( <i>int</i> point ) const
void	<i>set_color</i> ( <i>int</i> point, <i>Color</i> color )
<i>Color</i>	<i>get_color</i> ( <i>int</i> point ) const
<i>Color</i>	<i>interpolate</i> ( <i>float</i> offset )
<i>int</i>	<i>get_point_count</i> ( ) const
void	<i>set_offsets</i> ( <i>RealArray</i> offsets )
<i>RealArray</i>	<i>get_offsets</i> ( ) const
void	<i>set_colors</i> ( <i>ColorArray</i> colors )
<i>ColorArray</i>	<i>get_colors</i> ( ) const

### 9.59.3 Member Function Description

- void **add\_point** ( *float* offset, *Color* color )
- void **remove\_point** ( *int* offset )
- void **set\_offset** ( *int* point, *float* offset )
- *float* **get\_offset** ( *int* point ) const
- void **set\_color** ( *int* point, *Color* color )
- *Color* **get\_color** ( *int* point ) const
- *Color* **interpolate** ( *float* offset )
- *int* **get\_point\_count** ( ) const
- void **set\_offsets** ( *RealArray* offsets )
- *RealArray* **get\_offsets** ( ) const
- void **set\_colors** ( *ColorArray* colors )
- *ColorArray* **get\_colors** ( ) const

## 9.60 ConcavePolygonShape

Inherits: *Shape* < *Resource* < *Reference* < *Object*

Category: Core

### 9.60.1 Brief Description

Concave polygon shape.

## 9.60.2 Member Functions

void	<code>set_faces ( Vector3Array faces )</code>
<code>Vector3Array</code>	<code>get_faces () const</code>

## 9.60.3 Description

Concave polygon shape resource, which can be set into a [PhysicsBody](#) or area. This shape is created by feeding a list of triangles.

## 9.60.4 Member Function Description

- void `set_faces ( Vector3Array faces )`

Set the faces (an array of triangles).

- `Vector3Array get_faces () const`

Return the faces (an array of triangles).

## 9.61 ConcavePolygonShape2D

**Inherits:** [Shape2D](#) < [Resource](#) < [Reference](#) < [Object](#)

**Category:** Core

### 9.61.1 Brief Description

Concave polygon 2D shape resource for physics.

## 9.61.2 Member Functions

void	<code>set_segments ( Vector2Array segments )</code>
<code>Vector2Array</code>	<code>get_segments () const</code>

## 9.61.3 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 RigidBody 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.

## 9.61.4 Member Function Description

- void `set_segments` ( `Vector2Array` segments )

Set the array of segments.

- `Vector2Array get_segments () const`

Return the array of segments.

## 9.62 ConeTwistJoint

Inherits: `Joint < Spatial < Node < Object`

Category: Core

### 9.62.1 Brief Description

### 9.62.2 Member Functions

void	<code>set_param ( int param, float value )</code>
float	<code>get_param ( int param ) const</code>

### 9.62.3 Numeric Constants

- `PARAM_SWING_SPAN = 0`
- `PARAM_TWIST_SPAN = 1`
- `PARAM_BIAS = 2`
- `PARAM_SOFTNESS = 3`
- `PARAM_RELAXATION = 4`
- `PARAM_MAX = 5`

### 9.62.4 Member Function Description

- void `set_param ( int param, float value )`
- `float get_param ( int param ) const`

## 9.63 ConfigFile

Inherits: `Reference < Object`

Category: Core

### 9.63.1 Brief Description

### 9.63.2 Member Functions

void	<code>set_value ( String section, String key, var value )</code>
void	<code>get_value ( String section, String key, var default=NULL ) const</code>
<code>bool</code>	<code>has_section ( String section ) const</code>
<code>bool</code>	<code>has_section_key ( String section, String key ) const</code>
<code>StringArray</code>	<code>get_sections () const</code>
<code>StringArray</code>	<code>get_section_keys ( String section ) const</code>
Error	<code>load ( String path )</code>
Error	<code>save ( String path )</code>

### 9.63.3 Member Function Description

- void `set_value ( String section, String key, var value )`
- void `get_value ( String section, String key, var default=NULL ) const`
- `bool has_section ( String section ) const`
- `bool has_section_key ( String section, String key ) const`
- `StringArray get_sections () const`
- `StringArray get_section_keys ( String section ) const`
- Error `load ( String path )`
- Error `save ( String path )`

## 9.64 ConfirmationDialog

**Inherits:** `AcceptDialog < WindowDialog < Popup < Control < CanvasItem < Node < Object`

**Inherited By:** `EditorFileDialog, FileDialog`

**Category:** Core

### 9.64.1 Brief Description

Dialog for confirmation of actions.

### 9.64.2 Member Functions

<code>Button</code>	<code>get_cancel ()</code>
---------------------	----------------------------

### 9.64.3 Description

Dialog for confirmation of actions. This dialog inherits from `AcceptDialog`, but has by default an OK and Cancel button (in host OS order).

## 9.64.4 Member Function Description

- `Button get_cancel ()`

Return the cancel button.

# 9.65 Container

**Inherits:** `Control < CanvasItem < Node < Object`

**Inherited By:** `PanelContainer, GridContainer, ScrollContainer, MarginContainer, CenterContainer, GraphNode, SplitContainer, BoxContainer`

**Category:** Core

## 9.65.1 Brief Description

Base node for containers.

## 9.65.2 Member Functions

void	<code>queue_sort ()</code>
void	<code>fit_child_in_rect ( Control child, Rect2 rect )</code>

## 9.65.3 Signals

- `sort_children ()`

## 9.65.4 Numeric Constants

- `NOTIFICATION_SORT_CHILDREN = 50` — Notification for when sorting the children, it must be obeyed immediately.

## 9.65.5 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.

## 9.65.6 Member Function Description

- `void queue_sort ()`

Queue resort of the contained children. This is called automatically anyway, but can be called upon request.

- `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.

## 9.66 Control

**Inherits:** [CanvasItem](#) < [Node](#) < [Object](#)

**Inherited By:** [Label](#), [Tabs](#), [TextureFrame](#), [ButtonArray](#), [VideoPlayer](#), [LineEdit](#), [Container](#), [ReferenceFrame](#), [Patch9Frame](#), [TextEdit](#), [BaseButton](#), [Popup](#), [Tree](#), [Separator](#), [Panel](#), [TabContainer](#), [Range](#), [RichTextLabel](#), [GraphEdit](#), [ItemList](#)

**Category:** Core

### 9.66.1 Brief Description

Control is the base node for all the GUI components.

### 9.66.2 Member Functions

void	<code>_input_event ( InputEvent event ) virtual</code>
<code>bool</code>	<code>can_drop_data ( Vector2 pos, var data ) virtual</code>
void	<code>drop_data ( Vector2 pos, var data ) virtual</code>
<code>Object</code>	<code>get_drag_data ( Vector2 pos ) virtual</code>
<code>Vector2</code>	<code>get_minimum_size ( ) virtual</code>
void	<code>accept_event ()</code>
<code>Vector2</code>	<code>get_minimum_size ( ) const</code>
<code>Vector2</code>	<code>get_combined_minimum_size ( ) const</code>
void	<code>set_anchor ( int margin, int anchor_mode )</code>
<code>int</code>	<code>get_anchor ( int margin ) const</code>
void	<code>set_margin ( int margin, float offset )</code>
void	<code>set_anchor_and_margin ( int margin, int anchor_mode, float offset )</code>
void	<code>set_begin ( Vector2 pos )</code>
void	<code>set_end ( Vector2 pos )</code>
void	<code>set_pos ( Vector2 pos )</code>
void	<code>set_size ( Vector2 size )</code>
void	<code>set_custom_minimum_size ( Vector2 size )</code>
void	<code>set_global_pos ( Vector2 pos )</code>
void	<code>set_rotation ( float rotation )</code>
void	<code>set_scale ( Vector2 scale )</code>
<code>float</code>	<code>get_margin ( int margin ) const</code>
<code>Vector2</code>	<code>get_begin ( ) const</code>
<code>Vector2</code>	<code>get_end ( ) const</code>
<code>Vector2</code>	<code>get_pos ( ) const</code>
<code>Vector2</code>	<code>get_size ( ) const</code>
<code>float</code>	<code>get_rotation ( ) const</code>
<code>Vector2</code>	<code>get_scale ( ) const</code>
<code>Vector2</code>	<code>get_custom_minimum_size ( ) const</code>
<code>Vector2</code>	<code>get_parent_area_size ( ) const</code>
<code>Vector2</code>	<code>get_global_pos ( ) const</code>
<code>Rect2</code>	<code>get_rect ( ) const</code>
<code>Rect2</code>	<code>get_global_rect ( ) const</code>
void	<code>set_area_as_parent_rect ( int margin=0 )</code>
void	<code>show_modal ( bool exclusive=false )</code>

Continued on next page

Table 9.9 – continued from previous page

void	<code>set_focus_mode ( int mode )</code>
<i>bool</i>	<code>has_focus () const</code>
void	<code>grab_focus ()</code>
void	<code>release_focus ()</code>
<i>Control</i>	<code>get_focus_owner () const</code>
void	<code>set_h_size_flags ( int flags )</code>
<i>int</i>	<code>get_h_size_flags () const</code>
void	<code>set_stretch_ratio ( float ratio )</code>
<i>float</i>	<code>get_stretch_ratio () const</code>
void	<code>set_v_size_flags ( int flags )</code>
<i>int</i>	<code>get_v_size_flags () const</code>
void	<code>set_theme ( Theme theme )</code>
<i>Theme</i>	<code>get_theme () const</code>
void	<code>add_icon_override ( String name, Texture texture )</code>
void	<code>add_shader_override ( String name, Shader shader )</code>
void	<code>add_style_override ( String name, StyleBox stylebox )</code>
void	<code>add_font_override ( String name, Font font )</code>
void	<code>add_color_override ( String name, Color color )</code>
void	<code>add_constant_override ( String name, int constant )</code>
<i>Texture</i>	<code>get_icon ( String name, String type="" ) const</code>
<i>StyleBox</i>	<code>get_stylebox ( String name, String type="" ) const</code>
<i>Font</i>	<code>get_font ( String name, String type="" ) const</code>
<i>Color</i>	<code>get_color ( String name, String type="" ) const</code>
<i>int</i>	<code>get_constant ( String name, String type="" ) const</code>
<i>Control</i>	<code>get_parent_control () const</code>
void	<code>set_tooltip ( String tooltip )</code>
<i>String</i>	<code>get_tooltip ( Vector2 atpos=Vector2(0,0) ) const</code>
void	<code>set_default_cursor_shape ( int shape )</code>
<i>int</i>	<code>get_default_cursor_shape () const</code>
<i>int</i>	<code>get_cursor_shape ( Vector2 pos=Vector2(0,0) ) const</code>
void	<code>set_focus_neighbour ( int margin, NodePath neighbour )</code>
<i>NodePath</i>	<code>get_focus_neighbour ( int margin ) const</code>
void	<code>set_ignore_mouse ( bool ignore )</code>
<i>bool</i>	<code>is_ignoring_mouse () const</code>
void	<code>force_drag ( var data, Object preview )</code>
void	<code>set_stop_mouse ( bool stop )</code>
<i>bool</i>	<code>is_stopping_mouse () const</code>
void	<code>grab_click_focus ()</code>
void	<code>set_drag_preview ( Control control )</code>
void	<code>warp_mouse ( Vector2 to_pos )</code>

### 9.66.3 Signals

- `focus_enter ()`
- `mouse_enter ()`
- `resized ()`
- `minimum_size_changed ()`
- `size_flags_changed ()`

- `focus_exit()`
- `input_event( InputEvent ev )`
- `modal_close()`
- `mouse_exit()`

#### 9.66.4 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**
- **CURSOR\_CAN\_DROP = 7**
- **CURSOR\_FORBIDDEN = 8**
- **CURSOR\_VSIZE = 9**
- **CURSOR\_HSIZE = 10**
- **CURSOR\_BDIAGSIZE = 11**
- **CURSOR\_FDIAGSIZE = 12**
- **CURSOR\_MOVE = 13**
- **CURSOR\_VSPLIT = 14**

- **CURSOR\_HSPLIT = 15**
- **CURSOR\_HELP = 16**
- **SIZE\_EXPAND = 1**
- **SIZE\_FILL = 2**
- **SIZE\_EXPAND\_FILL = 3**

## 9.66.5 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 [Node.\\_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 overrided per each control by calling the [add\\_\\*\\_override](#) functions, or from the editor.

## 9.66.6 Member Function Description

- `void _input_event ( InputEvent event ) virtual`

Called when an input event reaches the control.

- `bool can_drop_data ( Vector2 pos, var data ) virtual`
- `void drop_data ( Vector2 pos, var data ) virtual`
- `Object get_drag_data ( Vector2 pos ) virtual`
- `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.

- `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](#).

- `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_combined_minimum_size () const`
- `void set_anchor ( int margin, int anchor_mode )`

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`).

- `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).

- `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_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_begin ( Vector2 pos )`

Sets MARGIN\_LEFT and MARGIN\_TOP at the same time. This is a helper (see `set_margin`).

- `void set_end ( Vector2 pos )`

Sets MARGIN\_RIGHT and MARGIN\_BOTTOM at the same time. This is a helper (see `set_margin`).

- `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_size ( Vector2 size )`

Changes MARGIN\_RIGHT and MARGIN\_BOTTOM to fit a given size. This is a helper (see `set_margin`).

- `void set_custom_minimum_size ( Vector2 size )`
- `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_rotation ( float rotation )`
- `void set_scale ( Vector2 scale )`
- `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_begin () const`
- `Vector2 get_end () const`

Returns MARGIN\_LEFT and MARGIN\_TOP at the same time. This is a helper (see `set_margin`).

- `Vector2 get_pos () const`

Returns the Control position, relative to the top-left corner of the parent Control and independent of the anchor mode.

- `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 :ref:`get\_minimum\_size<class\_Control\_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_rotation () const`
- `Vector2 get_scale () const`
- `Vector2 get_custom_minimum_size () const`
- `Vector2 get_parent_area_size () 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_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`).

- `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`).

- `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 show_modal ( bool exclusive=false )`

Display a Control as modal. Control must be a subwindow (see `set_as_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 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.

- `bool has_focus () const`

Return whether the Control is the current focused control (see `set_focus_mode`).

- `void grab_focus ()`

Steal the focus from another control and become the focused control (see `set_focus_mode`).

- `void release_focus ()`

Give up the focus, no other control will be able to receive keyboard input.

- `Control get_focus_owner () const`

Return which control is owning the keyboard focus, or null if no one.

- `void set_h_size_flags ( int flags )`

Hint for containers, set horizontal positioning flags.

- `int get_h_size_flags () const`

Hint for containers, return horizontal positioning flags.

- `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.

- `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.

- `void set_v_size_flags ( int flags )`

Hint for containers, set vertical positioning flags.

- `int get_v_size_flags ( ) const`

Hint for containers, return vertical positioning flags.

- `void set_theme ( Theme theme )`

Override whole the `Theme` for this Control and all its children controls.

- `Theme get_theme ( ) const`

Return a `Theme` override, if one exists (see `set_theme`).

- `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.

- `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_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.

- `Texture get_icon ( String name, String type="" ) const`

- `StyleBox get_stylebox ( String name, String type="" ) const`

- `Font get_font ( String name, String type="" ) const`

- `Color get_color ( String name, String type="" ) const`

- `int get_constant ( String name, String type="" ) const`

- `Control get_parent_control ( ) const`

- `void set_tooltip ( String tooltip )`

Set a tooltip, which will appear when the cursor is resting over this control.

- `String get_tooltip ( Vector2 atpos=Vector2(0,0) ) const`

Return the tooltip, which will appear when the cursor is resting over this control.

- `void set_default_cursor_shape ( int shape )`

Set the default cursor shape for this control. See enum CURSOR\_\* for the list of shapes.

- `int get_default_cursor_shape () const`

Return the default cursor shape for this control. See enum CURSOR\_\* for the list of shapes.

- `int get_cursor_shape ( Vector2 pos=Vector2(0,0) ) const`

Return the cursor shape at a certain position in the control.

- `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.

- `NodePath get_focus_neighbour ( int margin ) const`

Return the forced 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_ignore_mouse ( bool ignore )`

Ignore mouse events on this control (even touchpad events send mouse events).

- `bool is_ignoring_mouse () const`

Return if the control is ignoring mouse events (even touchpad events send mouse events).

- `void force_drag ( var data, Object preview )`
- `void set_stop_mouse ( bool stop )`
- `bool is_stopping_mouse () const`
- `void grab_click_focus ()`
- `void set_drag_preview ( Control control )`
- `void warp_mouse ( Vector2 to_pos )`

## 9.67 ConvexPolygonShape

**Inherits:** `Shape < Resource < Reference < Object`

**Category:** Core

### 9.67.1 Brief Description

Convex Polygon Shape.

### 9.67.2 Member Functions

<code>void</code>	<code>set_points ( Vector3Array points )</code>
<code>Vector3Array</code>	<code>get_points () const</code>

### 9.67.3 Description

Convex polygon shape resource, which can be set into a `PhysicsBody` or area.

## 9.67.4 Member Function Description

- void `set_points` ( `Vector3Array` points )
- `Vector3Array` `get_points` ( ) const

## 9.68 ConvexPolygonShape2D

**Inherits:** `Shape2D < Resource < Reference < Object`

**Category:** Core

### 9.68.1 Brief Description

Convex Polygon Shape for 2D physics.

### 9.68.2 Member Functions

void	<code>set_point_cloud</code> ( <code>Vector2Array</code> point_cloud )
void	<code>set_points</code> ( <code>Vector2Array</code> points )
<code>Vector2Array</code>	<code>get_points</code> ( ) const

### 9.68.3 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.

### 9.68.4 Member Function Description

- 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.

- `Vector2Array` `get_points` ( ) const

Return a list of points in either clockwise or counter clockwise order, forming a convex polygon.

## 9.69 CubeMap

**Inherits:** `Resource < Reference < Object`

**Category:** Core

### 9.69.1 Brief Description

### 9.69.2 Member Functions

<i>int</i>	<code>get_width () const</code>
<i>int</i>	<code>get_height () const</code>
<i>RID</i>	<code>get_rid () const</code>
<i>void</i>	<code>set_flags ( <i>int</i> flags )</code>
<i>int</i>	<code>get_flags () const</code>
<i>void</i>	<code>set_side ( <i>int</i> side, <i>Image</i> image )</code>
<i>Image</i>	<code>get_side ( <i>int</i> side ) const</code>
<i>void</i>	<code>set_storage ( <i>int</i> mode )</code>
<i>int</i>	<code>get_storage () const</code>
<i>void</i>	<code>set_lossy_storage_quality ( <i>float</i> quality )</code>
<i>float</i>	<code>get_lossy_storage_quality () const</code>

### 9.69.3 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**
- **FLAG\_FILTER = 4**
- **FLAGS\_DEFAULT = 7**

### 9.69.4 Member Function Description

- `int get_width () const`
- `int get_height () const`
- `RID get_rid () const`
- `void set_flags ( int flags )`
- `int get_flags () const`
- `void set_side ( int side, Image image )`
- `Image get_side ( int side ) const`
- `void set_storage ( int mode )`

- `int get_storage() const`
- `void set_lossy_storage_quality(float quality)`
- `float get_lossy_storage_quality() const`

## 9.70 Curve2D

**Inherits:** [Resource](#) < [Reference](#) < [Object](#)

**Category:** Core

### 9.70.1 Brief Description

Describes a Bezier curve in 2D space.

### 9.70.2 Member Functions

<code>int</code>	<code>get_point_count() const</code>
<code>void</code>	<code>add_point(Vector2 pos, Vector2 in=Vector2(0,0), Vector2 out=Vector2(0,0), int atpos=-1)</code>
<code>void</code>	<code>set_point_pos(int idx, Vector2 pos)</code>
<code>Vector2</code>	<code>get_point_pos(int idx) const</code>
<code>void</code>	<code>set_point_in(int idx, Vector2 pos)</code>
<code>Vector2</code>	<code>get_point_in(int idx) const</code>
<code>void</code>	<code>set_point_out(int idx, Vector2 pos)</code>
<code>Vector2</code>	<code>get_point_out(int idx) const</code>
<code>void</code>	<code>remove_point(int idx)</code>
<code>Vector2</code>	<code>interpolate(int idx, float t) const</code>
<code>Vector2</code>	<code>interpolatef(float fofs) const</code>
<code>void</code>	<code>set_bake_interval(float distance)</code>
<code>float</code>	<code>get_bake_interval() const</code>
<code>float</code>	<code>get_baked_length() const</code>
<code>Vector2</code>	<code>interpolate_baked(float offset, bool cubic=false) const</code>
<code>Vector2Array</code>	<code>get_baked_points() const</code>
<code>Vector2Array</code>	<code>tessellate(int max_stages=5, float tolerance_degrees=4) const</code>

### 9.70.3 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.

### 9.70.4 Member Function Description

- `int get_point_count() const`

Returns the number of points describing the curve.

- `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 ( $\text{atpos} < 0$  or  $\text{atpos} \geq \text{get\_point\_count}$ ), the point will be appended at the end of the point list.

- `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.

- `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).

- `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.

- `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).

- `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.

- `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).

- `void remove_point ( int idx )`

Deletes the point “idx” from the curve. Sends an error to the console if “idx” is out of bounds.

- `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  $(0.0 \geq t \leq 1)$  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 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 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.

- `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.

- `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).

- `Vector2Array get_baked_points ( ) const`

Returns the cache of points as a `Vector2Array`.

- `Vector2Array tessellate ( 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.

## 9.71 Curve3D

**Inherits:** `Resource < Reference < Object`

**Category:** Core

### 9.71.1 Brief Description

Describes a Bezier curve in 3D space.

## 9.71.2 Member Functions

<code>int</code>	<code>get_point_count () const</code>
<code>void</code>	<code>add_point ( Vector3 pos, Vector3 in=Vector3(0, 0, 0), Vector3 out=Vector3(0, 0, 0), int atpos=-1 )</code>
<code>void</code>	<code>set_point_pos ( int idx, Vector3 pos )</code>
<code>Vector3</code>	<code>get_point_pos ( int idx ) const</code>
<code>void</code>	<code>set_point_tilt ( int idx, float tilt )</code>
<code>float</code>	<code>get_point_tilt ( int idx ) const</code>
<code>void</code>	<code>set_point_in ( int idx, Vector3 pos )</code>
<code>Vector3</code>	<code>get_point_in ( int idx ) const</code>
<code>void</code>	<code>set_point_out ( int idx, Vector3 pos )</code>
<code>Vector3</code>	<code>get_point_out ( int idx ) const</code>
<code>void</code>	<code>remove_point ( int idx )</code>
<code>Vector3</code>	<code>interpolate ( int idx, float t ) const</code>
<code>Vector3</code>	<code>interpolatef ( float fofs ) const</code>
<code>void</code>	<code>set_bake_interval ( float distance )</code>
<code>float</code>	<code>get_bake_interval () const</code>
<code>float</code>	<code>get_baked_length () const</code>
<code>Vector3</code>	<code>interpolate_baked ( float offset, bool cubic=false ) const</code>
<code>Vector3Array</code>	<code>get_baked_points () const</code>
<code>RealArray</code>	<code>get_baked_tilts () const</code>
<code>Vector3Array</code>	<code>tesselate ( int max_stages=5, float tolerance_degrees=4 ) const</code>

## 9.71.3 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.

## 9.71.4 Member Function Description

- `int get_point_count () const`

Returns the number of points describing the curve.

- `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 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.

- `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)`.

- `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.

- `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.

- `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.

- `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).

- `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.

- `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).

- `void remove_point ( int idx )`

Deletes the point “idx” from the curve. Sends an error to the console if “idx” is out of bounds.

- `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 ( $0.0 \geq t \leq 1$ ) 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 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 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.

- `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.

- `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).

- `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`.

- `Vector3Array tessellate ( 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.

## 9.72 DampedSpringJoint2D

**Inherits:** `Joint2D < Node2D < CanvasItem < Node < Object`

**Category:** Core

### 9.72.1 Brief Description

Damped spring constraint for 2D physics.

### 9.72.2 Member Functions

<code>void</code>	<code>set_length ( float length )</code>
<code>float</code>	<code>get_length () const</code>
<code>void</code>	<code>set_rest_length ( float rest_length )</code>
<code>float</code>	<code>get_rest_length () const</code>
<code>void</code>	<code>set_stiffness ( float stiffness )</code>
<code>float</code>	<code>get_stiffness () const</code>
<code>void</code>	<code>set_damping ( float damping )</code>
<code>float</code>	<code>get_damping () const</code>

### 9.72.3 Description

Damped spring constraint for 2D physics. This resembles a spring joint that always want to go back to a given length.

## 9.72.4 Member Function Description

- void `set_length ( float length )`

Set the maximum length of the spring joint.

- `float get_length () const`

Return 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 this length when pulled apart.

- `float get_rest_length () const`

Return the resting length of the spring joint. The joint will always try to go back this length when pulled apart.

- void `set_stiffness ( float stiffness )`

Set the stiffness of the spring joint.

- `float get_stiffness () const`

Return the stiffness of the spring joint.

- void `set_damping ( float damping )`

Set the damping of the spring joint.

- `float get_damping () const`

Return the damping of the spring joint.

## 9.73 Dictionary

**Category:** Built-In Types

### 9.73.1 Brief Description

Dictionary type.

### 9.73.2 Member Functions

void	<code>clear ()</code>
<code>bool</code>	<code>empty ()</code>
void	<code>erase ( var value )</code>
<code>bool</code>	<code>has ( var value )</code>
<code>int</code>	<code>hash ()</code>
<code>Array</code>	<code>keys ()</code>
<code>int</code>	<code>parse_json ( String json )</code>
<code>int</code>	<code>size ()</code>
<code>String</code>	<code>to_json ()</code>

### 9.73.3 Description

Dictionary type. Associative container which contains values referenced by unique keys. Dictionaries are always passed by reference.

### 9.73.4 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 value )**

Erase a dictionary key/value pair by key.

- *bool* **has ( var value )**

Return true if the dictionary has a given key.

- *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 )**

- *int* **size ()**

Return the size of the dictionary (in pairs).

- *String* **to\_json ()**

## 9.74 DirectionalLight

**Inherits:** *Light < VisualInstance < Spatial < Node < Object*

**Category:** Core

### 9.74.1 Brief Description

Directional Light, such as the Sun or the Moon.

### 9.74.2 Member Functions

void	<i>set_shadow_mode ( int mode )</i>
<i>int</i>	<i>get_shadow_mode () const</i>
void	<i>set_shadow_param ( int param, float value )</i>
<i>float</i>	<i>get_shadow_param ( int param ) const</i>

### 9.74.3 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**

### 9.74.4 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.

### 9.74.5 Member Function Description

- `void set_shadow_mode ( int mode )`
- `int get_shadow_mode () const`
- `void set_shadow_param ( int param, float value )`
- `float get_shadow_param ( int param ) const`

## 9.75 Directory

Inherits: [Reference](#) < [Object](#)

Category: Core

### 9.75.1 Brief Description

### 9.75.2 Member Functions

Error	<code>open ( String path )</code>
<code>bool</code>	<code>list_dir_begin ()</code>
<code>String</code>	<code>get_next ()</code>
<code>bool</code>	<code>current_is_dir () const</code>
<code>void</code>	<code>list_dir_end ()</code>
<code>int</code>	<code>get_drive_count ()</code>
<code>String</code>	<code>get_drive ( int idx )</code>
Error	<code>change_dir ( String todir )</code>
<code>String</code>	<code>get_current_dir ()</code>
Error	<code>make_dir ( String name )</code>
Error	<code>make_dir_recursive ( String name )</code>
<code>bool</code>	<code>file_exists ( String name )</code>
<code>bool</code>	<code>dir_exists ( String name )</code>
<code>int</code>	<code>get_space_left ()</code>
Error	<code>copy ( String from, String to )</code>
Error	<code>rename ( String from, String to )</code>
Error	<code>remove ( String file )</code>

### 9.75.3 Member Function Description

- Error `open ( String path )`
- `bool` `list_dir_begin ()`
- `String` `get_next ()`
- `bool` `current_is_dir () const`
- `void` `list_dir_end ()`
- `int` `get_drive_count ()`
- `String` `get_drive ( int idx )`
- Error `change_dir ( String todir )`
- `String` `get_current_dir ()`
- Error `make_dir ( String name )`
- Error `make_dir_recursive ( String name )`
- `bool` `file_exists ( String name )`
- `bool` `dir_exists ( String name )`
- `int` `get_space_left ()`
- Error `copy ( String from, String to )`
- Error `rename ( String from, String to )`
- Error `remove ( String file )`

## 9.76 EditorFileDialog

**Inherits:** [ConfirmationDialog](#) < [AcceptDialog](#) < [WindowDialog](#) < [Popup](#) < [Control](#) < [CanvasItem](#) < [Node](#) < [Object](#)

**Category:** Core

### 9.76.1 Brief Description

### 9.76.2 Member Functions

void	<code>clear_filters ()</code>
void	<code>add_filter ( String filter )</code>
<i>String</i>	<code>get_current_dir () const</code>
<i>String</i>	<code>get_current_file () const</code>
<i>String</i>	<code>get_current_path () const</code>
void	<code>set_current_dir ( String dir )</code>
void	<code>set_current_file ( String file )</code>
void	<code>set_current_path ( String path )</code>
void	<code>set_mode ( int mode )</code>
<i>int</i>	<code>get_mode () const</code>
<i>VBoxContainer</i>	<code>get_vbox ()</code>
void	<code>set_access ( int access )</code>
<i>int</i>	<code>get_access () const</code>
void	<code>set_show_hidden_files ( bool show )</code>
<i>bool</i>	<code>isShowingHiddenFiles () const</code>
void	<code>set_display_mode ( int mode )</code>
<i>int</i>	<code>get_display_mode () const</code>
void	<code>invalidate ()</code>

### 9.76.3 Signals

- **files\_selected** ( *StringArray* paths )
- **dir\_selected** ( *String* dir )
- **file\_selected** ( *String* path )

### 9.76.4 Numeric Constants

- **MODE\_OPEN\_FILE** = 0
- **MODE\_OPEN\_FILES** = 1
- **MODE\_OPEN\_DIR** = 2
- **MODE\_SAVE\_FILE** = 3
- **ACCESS\_RESOURCES** = 0
- **ACCESS\_USERDATA** = 1
- **ACCESS\_FILESYSTEM** = 2

## 9.76.5 Member Function Description

- void **clear\_filters** ()
- void **add\_filter** ( *String* filter )
- *String* **get\_current\_dir** ( ) const
- *String* **get\_current\_file** ( ) const
- *String* **get\_current\_path** ( ) const
- void **set\_current\_dir** ( *String* dir )
- void **set\_current\_file** ( *String* file )
- void **set\_current\_path** ( *String* path )
- void **set\_mode** ( *int* mode )
- *int* **get\_mode** ( ) const
- *VBoxContainer* **get\_vbox** ( )
- void **set\_access** ( *int* access )
- *int* **get\_access** ( ) const
- void **set\_show\_hidden\_files** ( *bool* show )
- *bool* **isShowingHiddenFiles** ( ) const
- void **set\_display\_mode** ( *int* mode )
- *int* **get\_display\_mode** ( ) const
- void **invalidate** ( )

## 9.77 EditorImportPlugin

Inherits: *Reference* < *Object*

Category: Core

### 9.77.1 Brief Description

### 9.77.2 Member Functions

<i>RawArray</i>	<i>custom_export</i> ( <i>String</i> path ) virtual
<i>String</i>	<i>get_name</i> ( ) virtual
<i>String</i>	<i>getVisibleName</i> ( ) virtual
<i>int</i>	<i>import</i> ( <i>String</i> path, ResourceImportMetaData from ) virtual
void	<i>importDialog</i> ( <i>String</i> from ) virtual

### 9.77.3 Member Function Description

- *RawArray* **custom\_export** ( *String* path ) virtual
- *String* **get\_name** ( ) virtual

- `String get_visible_name() virtual`
- `int import ( String path, ResourceImportMetaData from ) virtual`
- `void import_dialog ( String from ) virtual`

## 9.78 EditorPlugin

**Inherits:** `Node < Object`

**Category:** Core

### 9.78.1 Brief Description

### 9.78.2 Member Functions

<code>void</code>	<code>apply_changes () virtual</code>
<code>void</code>	<code>clear () virtual</code>
<code>void</code>	<code>edit ( Object object ) virtual</code>
<code>bool</code>	<code>forward_input_event ( InputEvent event ) virtual</code>
<code>bool</code>	<code>forward_spatial_input_event ( Camera camera, InputEvent event ) virtual</code>
<code>StringArray</code>	<code>get_breakpoints () virtual</code>
<code>String</code>	<code>get_name () virtual</code>
<code>Dictionary</code>	<code>get_state () virtual</code>
<code>bool</code>	<code>handles ( Object object ) virtual</code>
<code>bool</code>	<code>has_main_screen () virtual</code>
<code>void</code>	<code>make_visible ( bool visible ) virtual</code>
<code>void</code>	<code>set_state ( Dictionary state ) virtual</code>
<code>Object</code>	<code>get_undo_redo ()</code>
<code>void</code>	<code>add_custom_control ( int container, Object control )</code>
<code>void</code>	<code>add_custom_type ( String type, String base, Script script, Texture icon )</code>
<code>void</code>	<code>remove_custom_type ( String type )</code>

### 9.78.3 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`

### 9.78.4 Member Function Description

- `void apply_changes () virtual`
- `void clear () virtual`
- `void edit ( Object object ) virtual`

- `bool forward_input_event ( InputEvent event ) virtual`
- `bool forward_spatial_input_event ( Camera camera, InputEvent event ) virtual`
- `StringArray get_breakpoints () virtual`
- `String get_name () virtual`
- `Dictionary get_state () virtual`
- `bool handles ( Object object ) virtual`
- `bool has_main_screen () virtual`
- `void make_visible ( bool visible ) virtual`
- `void set_state ( Dictionary state ) virtual`
- `Object get_undo_redo ()`
- `void add_custom_control ( int container, Object control )`
- `void add_custom_type ( String type, String base, Script script, Texture icon )`
- `void remove_custom_type ( String type )`

## 9.79 EditorScenePostImport

Inherits: *Reference* < *Object*

Category: Core

### 9.79.1 Brief Description

### 9.79.2 Member Functions

void	<code>post_import ( Object scene ) virtual</code>
------	---

### 9.79.3 Member Function Description

- `void post_import ( Object scene ) virtual`

## 9.80 EditorScript

Inherits: *Reference* < *Object*

Category: Core

### 9.80.1 Brief Description

### 9.80.2 Member Functions

void	<code>_run () virtual</code>
void	<code>add_root_node ( Object node )</code>
<i>Object</i>	<code>get_scene ()</code>

### 9.80.3 Member Function Description

- void `_run()` virtual
- void `add_root_node( Object node )`
- `Object get_scene()`

## 9.81 Environment

**Inherits:** `Resource < Reference < Object`

**Category:** Core

### 9.81.1 Brief Description

### 9.81.2 Member Functions

void	<code>set_background( int bgmode )</code>
<code>int</code>	<code>get_background() const</code>
void	<code>set_background_param( int param, var value )</code>
void	<code>get_background_param( int param ) const</code>
void	<code>set_enable_fx( int effect, bool enabled )</code>
<code>bool</code>	<code>is_fx_enabled( int effect ) const</code>
void	<code>fx_set_param( int param, var value )</code>
void	<code>fx_get_param( int param ) const</code>

### 9.81.3 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\_THRESHOLD** = 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\_THRESHOLD** = 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

## 9.81.4 Member Function Description

- void **set\_background** ( *int* bgmode )
- *int* **get\_background** ( ) const
- void **set\_background\_param** ( *int* param, var value )
- void **get\_background\_param** ( *int* param ) const
- void **set\_enable\_fx** ( *int* effect, *bool* enabled )
- *bool* **is\_fx\_enabled** ( *int* effect ) const
- void **fx\_set\_param** ( *int* param, var value )
- void **fx\_get\_param** ( *int* param ) const

## 9.82 EventPlayer

Inherits: *Node* < *Object*

Category: Core

### 9.82.1 Brief Description

Class for event stream playback.

## 9.82.2 Member Functions

void	<code>set_stream ( EventStream stream )</code>
<i>EventStream</i>	<code>get_stream () const</code>
void	<code>play ()</code>
void	<code>stop ()</code>
<i>bool</i>	<code>is_playing () const</code>
void	<code>set_paused ( bool paused )</code>
<i>bool</i>	<code>is_paused () const</code>
void	<code>set_loop ( bool enabled )</code>
<i>bool</i>	<code>has_loop () const</code>
void	<code>set_volume ( float volume )</code>
<i>float</i>	<code>get_volume () const</code>
void	<code>set_pitch_scale ( float pitch_scale )</code>
<i>float</i>	<code>get_pitch_scale () const</code>
void	<code>set_tempo_scale ( float tempo_scale )</code>
<i>float</i>	<code>get_tempo_scale () const</code>
void	<code>set_volume_db ( float db )</code>
<i>float</i>	<code>get_volume_db () const</code>
<i>String</i>	<code>get_stream_name () const</code>
<i>int</i>	<code>get_loop_count () const</code>
<i>float</i>	<code>get_pos () const</code>
void	<code>seek_pos ( float time )</code>
<i>float</i>	<code>get_length () const</code>
void	<code>set_autoplay ( bool enabled )</code>
<i>bool</i>	<code>has_autoplay () const</code>
void	<code>set_channel_volume ( int channel, float channel_volume )</code>
<i>float</i>	<code>get_channel_volume ( int channel ) const</code>
<i>float</i>	<code>get_channel_last_note_time ( int channel ) const</code>

## 9.82.3 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.

## 9.82.4 Member Function Description

- void `set_stream ( EventStream stream )`

Set the *EventStream* this player will play.

- *EventStream* `get_stream () const`

Return the currently assigned stream.

- void `play ()`

Play the currently assigned stream.

- void `stop ()`

Stop playing.

- `bool is_playing () const`

Return whether this player is playing.

- `void set_paused ( bool paused )`

Pause stream playback.

- `bool is_paused () const`

Return whether the playback is currently paused.

- `void set_loop ( bool enabled )`

Set whether the stream will be restarted at the end.

- `bool has_loop () const`

Return whether this player will be restart the playback at the end.

- `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.

- `float get_volume () const`

Return the playback volume for this player.

- `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.

- `float get_pitch_scale () const`

Return the pitch scale factor for this player.

- `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.

- `float get_tempo_scale () const`

Return the tempo multiplier.

- `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.

- `float get_volume_db () const`

Return the playback volume for this player, in decibels.

- `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>".

- `int get_loop_count () const`

Return the number of times the playback has looped.

- `float get_pos () const`

Return the playback position. May be in seconds, but depends on the stream type.

- `void seek_pos (float time)`

Set the playback position. May be in seconds, but depends on the stream type.

- `float get_length () const`

Return the song length. 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.

- `bool has_autoplay () const`

Return 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.

- `float get_channel_volume (int channel) const`

Return the volume scale for an individual channel of the stream.

- `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.

## 9.83 EventStream

**Inherits:** `Resource < Reference < Object`

**Inherited By:** `EventStreamChibi`

**Category:** Core

### 9.83.1 Brief Description

Base class for all event-based stream drivers.

### 9.83.2 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.

## 9.84 EventStreamChibi

**Inherits:** `EventStream < Resource < Reference < Object`

**Category:** Core

## 9.84.1 Brief Description

Driver for MOD playback.

## 9.84.2 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.

## 9.85 File

**Inherits:** [Reference](#) < [Object](#)

**Category:** Core

### 9.85.1 Brief Description

### 9.85.2 Member Functions

<code>int</code>	<code>open_encrypted ( String path, int mode_flags, RawArray key )</code>
<code>int</code>	<code>open_encrypted_with_pass ( String path, int mode_flags, String pass )</code>
<code>int</code>	<code>open ( String path, int flags )</code>
<code>void</code>	<code>close ()</code>
<code>bool</code>	<code>is_open () const</code>
<code>void</code>	<code>seek ( int pos )</code>
<code>void</code>	<code>seek_end ( int pos=0 )</code>
<code>int</code>	<code>get_pos () const</code>
<code>int</code>	<code>get_len () const</code>
<code>bool</code>	<code>eof_reached () const</code>
<code>int</code>	<code>get_8 () const</code>
<code>int</code>	<code>get_16 () const</code>
<code>int</code>	<code>get_32 () const</code>
<code>int</code>	<code>get_64 () const</code>
<code>float</code>	<code>get_float () const</code>
<code>float</code>	<code>get_double () const</code>
<code>float</code>	<code>get_real () const</code>
<code>RawArray</code>	<code>get_buffer ( int len ) const</code>
<code>String</code>	<code>get_line () const</code>
<code>String</code>	<code>get_as_text () const</code>
<code>bool</code>	<code>get_endian_swap ()</code>
<code>void</code>	<code>set_endian_swap ( bool enable )</code>

Continued on next page

Table 9.10 – continued from previous page

Error	<code>get_error() const</code>
void	<code>get_var() const</code>
<code>StringArray</code>	<code>get_csv_line( String delim="," ) const</code>
void	<code>store_8( int value )</code>
void	<code>store_16( int value )</code>
void	<code>store_32( int value )</code>
void	<code>store_64( int value )</code>
void	<code>store_float( float value )</code>
void	<code>store_double( float value )</code>
void	<code>store_real( float value )</code>
void	<code>store_buffer( RawArray buffer )</code>
void	<code>store_line( String line )</code>
void	<code>store_string( String string )</code>
void	<code>store_var( var value )</code>
void	<code>store_pascal_string( String string )</code>
<code>String</code>	<code>get_pascal_string()</code>
<code>bool</code>	<code>file_exists( String path ) const</code>

### 9.85.3 Numeric Constants

- **READ = 1**
- **WRITE = 2**
- **READ\_WRITE = 3**
- **WRITE\_READ = 7**

### 9.85.4 Member Function Description

- `int open_encrypted( String path, int mode_flags, RawArray key )`
- `int open_encrypted_with_pass( String path, int mode_flags, String pass )`
- `int open( String path, int flags )`
- `void close()`
- `bool is_open() const`
- `void seek( int pos )`
- `void seek_end( int pos=0 )`
- `int get_pos() const`
- `int get_len() const`
- `bool eof_reached() const`
- `int get_8() const`
- `int get_16() const`
- `int get_32() const`
- `int get_64() const`
- `float get_float() const`

- `float get_double()` const
- `float get_real()` const
- `RawArray get_buffer(int len)` const
- `String get_line()` const
- `String get_as_text()` const
- `bool get_endian_swap()`
- `void set_endian_swap(bool enable)`
- Error `get_error()` const
- void `get_var()` const
- `StringArray get_csv_line(String delim=",")` const
- void `store_8(int value)`
- void `store_16(int value)`
- void `store_32(int value)`
- void `store_64(int value)`
- void `store_float(float value)`
- void `store_double(float value)`
- void `store_real(float value)`
- void `store_buffer(RawArray buffer)`
- void `store_line(String line)`
- void `store_string(String string)`
- void `store_var(var value)`
- void `store_pascal_string(String string)`
- `String get_pascal_string()`
- `bool file_exists(String path)` const

## 9.86 FileDialog

Inherits: `ConfirmationDialog < AcceptDialog < WindowDialog < Popup < Control < CanvasItem < Node < Object`

Category: Core

### 9.86.1 Brief Description

Dialog for selecting files or directories in the filesystem.

## 9.86.2 Member Functions

void	<code>clear_filters()</code>
void	<code>add_filter( String filter )</code>
<i>String</i>	<code>get_current_dir() const</code>
<i>String</i>	<code>get_current_file() const</code>
<i>String</i>	<code>get_current_path() const</code>
void	<code>set_current_dir( String dir )</code>
void	<code>set_current_file( String file )</code>
void	<code>set_current_path( String path )</code>
void	<code>set_mode( int mode )</code>
<i>int</i>	<code>get_mode() const</code>
<i>VBoxContainer</i>	<code>get_ybox()</code>
void	<code>set_access( int access )</code>
<i>int</i>	<code>get_access() const</code>
void	<code>set_show_hidden_files( bool show )</code>
<i>bool</i>	<code>isShowingHiddenFiles() const</code>
void	<code>invalidate()</code>

## 9.86.3 Signals

- `files_selected( StringArray paths )`
- `dir_selected( String dir )`
- `file_selected( String path )`

## 9.86.4 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_SAVE_FILE = 3` — The dialog will warn when a file exists.
- `ACCESS_RESOURCES = 0`
- `ACCESS_USERDATA = 1`
- `ACCESS_FILESYSTEM = 2`

## 9.86.5 Description

FileDialog is a preset dialog used to choose files and directories in the filesystem. It supports filter masks.

## 9.86.6 Member Function Description

- void `clear_filters()`

Clear all the added filters in the dialog.

- void `add_filter( String filter )`

Add a custom filter. Filter format is: “mask ; description”, example (C++): dialog->add\_filter(“\*.png ; PNG Images”);

- `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).

- `void set_current_dir ( String dir )`
- `void set_current_file ( String file )`
- `void set_current_path ( String path )`
- `void set_mode ( int mode )`

Set the file dialog mode from the MODE\_\* enum.

- `int get_mode () const`

Get the file dialog mode from the MODE\_\* enum.

- `VBoxContainer get_vbox ()`
- `void set_access ( int access )`
- `int get_access () const`
- `void set_show_hidden_files ( bool show )`
- `bool isShowingHiddenFiles () const`
- `void invalidate ()`

## 9.87 FixedMaterial

**Inherits:** `Material < Resource < Reference < Object`

**Category:** Core

### 9.87.1 Brief Description

Simple Material with a fixed parameter set.

## 9.87.2 Member Functions

void	<code>set_parameter ( int param, var value )</code>
void	<code>get_parameter ( int param ) const</code>
void	<code>set_texture ( int param, Texture texture )</code>
<i>Texture</i>	<code>get_texture ( int param ) const</code>
void	<code>set_texcoord_mode ( int param, int mode )</code>
<i>int</i>	<code>get_texcoord_mode ( int param ) const</code>
void	<code>set_fixed_flag ( int flag, bool value )</code>
<i>bool</i>	<code>get_fixed_flag ( int flag ) const</code>
void	<code>set_uv_transform ( Transform transform )</code>
<i>Transform</i>	<code>get_uv_transform ( ) const</code>
void	<code>set_light_shader ( int shader )</code>
<i>int</i>	<code>get_light_shader ( ) const</code>
void	<code>set_point_size ( float size )</code>
<i>float</i>	<code>get_point_size ( ) const</code>

## 9.87.3 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\_SPHERE = 3**
- **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.
- **FLAG\_USE\_ALPHA = 0**
- **FLAG\_USE\_COLOR\_ARRAY = 1**
- **FLAG\_USE\_POINT\_SIZE = 2**
- **FLAG\_DISCARD\_ALPHA = 3**
- **LIGHT\_SHADER\_LAMBERT = 0**
- **LIGHT\_SHADER\_WRAP = 1**
- **LIGHT\_SHADER\_VELVET = 2**
- **LIGHT\_SHADER\_TOON = 3**

## 9.87.4 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.

## 9.87.5 Member Function Description

- `void set_parameter ( int param, var 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 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.

- `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](#)).

- `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](#)).

- `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.

- `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.

- `void set_fixed_flag ( int flag, bool value )`

- `bool get_fixed_flag ( int flag ) const`

- `void set_uv_transform ( Transform transform )`

Sets a special transform used to post-transform UV coordinates of the uv\_xfrom tecoord mode: TEXCOORD\_UV\_TRANSFORM.

- `Transform get_uv_transform ( ) const`

Returns the special transform used to post-transform UV coordinates of the uv\_xfrom tecoord mode: TEXCOORD\_UV\_TRANSFORM.

- `void set_light_shader ( int shader )`

- `int get_light_shader ( ) const`

- `void set_point_size ( float size )`

- `float get_point_size ( ) const`

## 9.88 float

**Category:** Built-In Types

### 9.88.1 Brief Description

### 9.88.2 Member Functions

<i>float</i>	<code>float ( <i>bool</i> from )</code>
<i>float</i>	<code>float ( <i>int</i> from )</code>
<i>float</i>	<code>float ( <i>String</i> from )</code>

### 9.88.3 Member Function Description

- `float float ( bool from )`
- `float float ( int from )`
- `float float ( String from )`

## 9.89 Font

**Inherits:** *Resource* < *Reference* < *Object*

**Category:** Core

### 9.89.1 Brief Description

Internationalized font and text drawing support.

## 9.89.2 Member Functions

<code>int</code>	<code>create_from_fnt ( String path )</code>
<code>void</code>	<code>set_height ( float px )</code>
<code>float</code>	<code>get_height () const</code>
<code>void</code>	<code>set_ascent ( float px )</code>
<code>float</code>	<code>get_ascent () const</code>
<code>float</code>	<code>get_descent () const</code>
<code>void</code>	<code>add_kerning_pair ( int char_a, int char_b, int kerning )</code>
<code>int</code>	<code>get_kerning_pair ( int char_a, int char_b ) const</code>
<code>void</code>	<code>add_texture ( Texture texture )</code>
<code>void</code>	<code>add_char ( int character, int texture, Rect2 rect, Vector2 align=Vector2(0,0), float advance=-1 )</code>
<code>int</code>	<code>get_texture_count () const</code>
<code>Texture</code>	<code>get_texture ( int idx ) const</code>
<code>Vector2</code>	<code>get_char_size ( int char, int next=0 ) const</code>
<code>Vector2</code>	<code>get_string_size ( String string ) const</code>
<code>void</code>	<code>set_distance_field_hint ( bool enable )</code>
<code>bool</code>	<code>is_distance_field_hint () const</code>
<code>void</code>	<code>clear ()</code>
<code>void</code>	<code>draw ( RID canvas_item, Vector2 pos, String string, Color modulate=Color(1,1,1,1), int clip_w=-1 ) const</code>
<code>float</code>	<code>draw_char ( RID canvas_item, Vector2 pos, int char, int next=-1, Color modulate=Color(1,1,1,1) ) const</code>
<code>void</code>	<code>set_fallback ( Object fallback )</code>
<code>Object</code>	<code>get_fallback () const</code>

## 9.89.3 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.

## 9.89.4 Member Function Description

- `int create_from_fnt ( String path )`
- `void set_height ( float px )`

Set the total font height (ascent plus descent) in pixels.

- `float get_height () const`

Return the total font height (ascent plus descent) in pixels.

- `void set_ascent ( float px )`

Set the font ascent (number of pixels above the baseline).

- `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).

- `void add_kerning_pair ( int char_a, int char_b, int kerning )`

Add a kerning pair to the *Font* as a difference. Kerning pairs are special cases where a typeface advance is determined by the next character.

- `int get_kerning_pair ( int char_a, int char_b ) const`

Return a kerning pair 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 *Font*.

- `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.

- `int get_texture_count ( ) const`
- `Texture get_texture ( int idx ) const`
- `Vector2 get_char_size ( int char, int next=0 ) const`

Return the size of a character, optionally taking kerning into account if the next character is provided.

- `Vector2 get_string_size ( String string ) const`

Return the size of a string, taking kerning and advance into account.

- `void set_distance_field_hint ( bool enable )`
- `bool is_distance_field_hint ( ) const`
- `void clear ( )`

Clear all the font data.

- `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.

- `void set_fallback ( Object fallback )`
- `Object get_fallback ( ) const`

## 9.90 FuncRef

Inherits: *Reference* < *Object*

Category: Core

## 9.90.1 Brief Description

## 9.90.2 Member Functions

void	<code>call_func</code> ( var arg0=NULL, var arg1=NULL, var arg2=NULL, var arg3=NULL, var arg4=NULL, var arg5=NULL, var arg6=NULL, var arg7=NULL, var arg8=NULL, var arg9=NULL )
void	<code>set_instance</code> ( <i>Object</i> instance )
void	<code>set_function</code> ( <i>String</i> name )

## 9.90.3 Member Function Description

- void `call_func` ( var arg0=NULL, var arg1=NULL, var arg2=NULL, var arg3=NULL, var arg4=NULL, var arg5=NULL, var arg6=NULL, var arg7=NULL, var arg8=NULL, var arg9=NULL )
- void `set_instance` ( *Object* instance )
- void `set_function` ( *String* name )

## 9.91 GDFunctionState

Inherits: *Reference* < *Object*

Category: Core

## 9.91.1 Brief Description

## 9.91.2 Member Functions

Variant	<code>resume</code> ( var arg=NULL )
<i>bool</i>	<code>is_valid</code> () const

## 9.91.3 Member Function Description

- Variant `resume` ( var arg=NULL )
- *bool* `is_valid` () const

## 9.92 GDNativeClass

Inherits: *Reference* < *Object*

Category: Core

## 9.92.1 Brief Description

## 9.92.2 Member Functions

void	<code>new</code> ()
------	---------------------

### 9.92.3 Member Function Description

- void **new ()**

## 9.93 GDScript

**Inherits:** *Script < Resource < Reference < Object*

**Category:** Core

### 9.93.1 Brief Description

### 9.93.2 Member Functions

void	<i>new ()</i>
<i>RawArray</i>	<i>get_as_byte_code () const</i>

### 9.93.3 Member Function Description

- void **new ()**
- *RawArray* **get\_as\_byte\_code () const**

## 9.94 Generic6DOFJoint

**Inherits:** *Joint < Spatial < Node < Object*

**Category:** Core

### 9.94.1 Brief Description

### 9.94.2 Member Functions

void	<i>set_param_x ( int param, float value )</i>
<i>float</i>	<i>get_param_x ( int param ) const</i>
void	<i>set_param_y ( int param, float value )</i>
<i>float</i>	<i>get_param_y ( int param ) const</i>
void	<i>set_param_z ( int param, float value )</i>
<i>float</i>	<i>get_param_z ( int param ) const</i>
void	<i>set_flag_x ( int flag, bool value )</i>
<i>bool</i>	<i>get_flag_x ( int flag ) const</i>
void	<i>set_flag_y ( int flag, bool value )</i>
<i>bool</i>	<i>get_flag_y ( int flag ) const</i>
void	<i>set_flag_z ( int flag, bool value )</i>
<i>bool</i>	<i>get_flag_z ( int flag ) const</i>

### 9.94.3 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**

### 9.94.4 Member Function Description

- void **set\_param\_x** ( *int* param, *float* value )
- *float* **get\_param\_x** ( *int* param ) const
- void **set\_param\_y** ( *int* param, *float* value )
- *float* **get\_param\_y** ( *int* param ) const
- void **set\_param\_z** ( *int* param, *float* value )
- *float* **get\_param\_z** ( *int* param ) const
- void **set\_flag\_x** ( *int* flag, *bool* value )
- *bool* **get\_flag\_x** ( *int* flag ) const
- void **set\_flag\_y** ( *int* flag, *bool* value )
- *bool* **get\_flag\_y** ( *int* flag ) const
- void **set\_flag\_z** ( *int* flag, *bool* value )
- *bool* **get\_flag\_z** ( *int* flag ) const

## 9.95 Geometry

Inherits: *Object*

Category: Core

### 9.95.1 Brief Description

### 9.95.2 Member Functions

<i>Array</i>	<code>build_box_planes ( Vector3 extents )</code>
<i>Array</i>	<code>build_cylinder_planes ( float radius, float height, int sides, int axis=2 )</code>
<i>Array</i>	<code>build_capsule_planes ( float radius, float height, int sides, int lats, int axis=2 )</code>
<i>float</i>	<code>segment_intersects_circle ( Vector2 segment_from, Vector2 segment_to, Vector2 circle_pos, float circle_radius )</code>
<i>void</i>	<code>segment_intersects_segment_2d ( Vector2 from_a, Vector2 to_a, Vector2 from_b, Vector2 to_b )</code>
<i>Vector2Array</i>	<code>get_closest_points_between_segments_2d ( Vector2 p1, Vector2 q1, Vector2 p2, Vector2 q2 )</code>
<i>Vector3Array</i>	<code>get_closest_points_between_segments ( Vector3 p1, Vector3 p2, Vector3 q1, Vector3 q2 )</code>
<i>Vector3</i>	<code>get_closest_point_to_segment ( Vector3 point, Vector3 s1, Vector3 s2 )</code>
<i>int</i>	<code>get_uv84_normal_bit ( Vector3 normal )</code>
<i>void</i>	<code>ray_intersects_triangle ( Vector3 from, Vector3 dir, Vector3 a, Vector3 b, Vector3 c )</code>
<i>void</i>	<code>segment_intersects_triangle ( Vector3 from, Vector3 to, Vector3 a, Vector3 b, Vector3 c )</code>
<i>Vector3Array</i>	<code>segment_intersects_sphere ( Vector3 from, Vector3 to, Vector3 spos, float sradius )</code>
<i>Vector3Array</i>	<code>segment_intersects_cylinder ( Vector3 from, Vector3 to, float height, float radius )</code>
<i>Vector3Array</i>	<code>segment_intersects_convex ( Vector3 from, Vector3 to, Array planes )</code>
<i>bool</i>	<code>point_is_inside_triangle ( Vector2 point, Vector2 a, Vector2 b, Vector2 c ) const</code>
<i>IntArray</i>	<code>triangulate_polygon ( Vector2Array polygon )</code>
<i>Dictionary</i>	<code>make_atlas ( Vector2Array sizes )</code>

### 9.95.3 Member Function Description

- *Array* `build_box_planes ( Vector3 extents )`
- *Array* `build_cylinder_planes ( float radius, float height, int sides, int axis=2 )`
- *Array* `build_capsule_planes ( float radius, float height, int sides, int lats, int axis=2 )`
- *float* `segment_intersects_circle ( Vector2 segment_from, Vector2 segment_to, Vector2 circle_pos, float circle_radius )`
- *void* `segment_intersects_segment_2d ( Vector2 from_a, Vector2 to_a, Vector2 from_b, Vector2 to_b )`
- *Vector2Array* `get_closest_points_between_segments_2d ( Vector2 p1, Vector2 q1, Vector2 p2, Vector2 q2 )`
- *Vector3Array* `get_closest_points_between_segments ( Vector3 p1, Vector3 p2, Vector3 q1, Vector3 q2 )`
- *Vector3* `get_closest_point_to_segment ( Vector3 point, Vector3 s1, Vector3 s2 )`
- *int* `get_uv84_normal_bit ( Vector3 normal )`
- *void* `ray_intersects_triangle ( Vector3 from, Vector3 dir, Vector3 a, Vector3 b, Vector3 c )`

- void **segment\_intersects\_triangle** ( *Vector3* from, *Vector3* to, *Vector3* a, *Vector3* b, *Vector3* c )
- *Vector3Array* **segment\_intersects\_sphere** ( *Vector3* from, *Vector3* to, *Vector3* spos, *float* sradius )
- *Vector3Array* **segment\_intersects\_cylinder** ( *Vector3* from, *Vector3* to, *float* height, *float* radius )
- *Vector3Array* **segment\_intersects\_convex** ( *Vector3* from, *Vector3* to, *Array* planes )
- *bool* **point\_is\_inside\_triangle** ( *Vector2* point, *Vector2* a, *Vector2* b, *Vector2* c ) const
- *IntArray* **triangulate\_polygon** ( *Vector2Array* polygon )
- *Dictionary* **make\_atlas** ( *Vector2Array* sizes )

## 9.96 GeometryInstance

**Inherits:** *VisualInstance* < *Spatial* < *Node* < *Object*

**Inherited By:** *MultiMeshInstance*, *SpriteBase3D*, *MeshInstance*, *Particles*, *Quad*, *TestCube*, *ImmediateGeometry*

**Category:** Core

### 9.96.1 Brief Description

Base node for geometry based visual instances.

### 9.96.2 Member Functions

void	<i>set_material_override</i> ( <i>Object</i> material )
<i>Object</i>	<i>get_material_override</i> () const
void	<i>set_flag</i> ( <i>int</i> flag, <i>bool</i> value )
<i>bool</i>	<i>get_flag</i> ( <i>int</i> flag ) const
void	<i>set_draw_range_begin</i> ( <i>float</i> mode )
<i>float</i>	<i>get_draw_range_begin</i> () const
void	<i>set_draw_range_end</i> ( <i>float</i> mode )
<i>float</i>	<i>get_draw_range_end</i> () const
void	<i>set_baked_light_texture_id</i> ( <i>int</i> id )
<i>int</i>	<i>get_baked_light_texture_id</i> () const
void	<i>set_extra_cull_margin</i> ( <i>float</i> margin )
<i>float</i>	<i>get_extra_cull_margin</i> () const

### 9.96.3 Numeric Constants

- **FLAG\_VISIBLE** = 0
- **FLAG\_CAST\_SHADOW** = 3
- **FLAG\_RECEIVE\_SHADOWS** = 4
- **FLAG\_BILLBOARD** = 1
- **FLAG\_BILLBOARD\_FIX\_Y** = 2
- **FLAG\_DEPTH\_SCALE** = 5
- **FLAG\_VISIBLE\_IN\_ALL\_ROOMS** = 6

- **FLAG\_MAX = 8**

## 9.96.4 Description

Base node for geometry based visual instances. Shares some common functionality like visibility and custom materials.

## 9.96.5 Member Function Description

- void **set\_material\_override** ( *Object* material )

Set the material override for the whole geometry.

- *Object* **get\_material\_override** ( ) const

Return the material override for the whole geometry.

- void **set\_flag** ( *int* flag, *bool* value )
- *bool* **get\_flag** ( *int* flag ) const
- void **set\_draw\_range\_begin** ( *float* mode )
- *float* **get\_draw\_range\_begin** ( ) const
- void **set\_draw\_range\_end** ( *float* mode )
- *float* **get\_draw\_range\_end** ( ) const
- void **set\_baked\_light\_texture\_id** ( *int* id )
- *int* **get\_baked\_light\_texture\_id** ( ) const
- void **set\_extra\_cull\_margin** ( *float* margin )
- *float* **get\_extra\_cull\_margin** ( ) const

## 9.97 Globals

**Inherits:** *Object*

**Category:** Core

### 9.97.1 Brief Description

Contains global variables accessible from everywhere.

## 9.97.2 Member Functions

<code>bool</code>	<code>has ( String name ) const</code>
<code>void</code>	<code>set_order ( String name, int pos )</code>
<code>int</code>	<code>get_order ( String name ) const</code>
<code>void</code>	<code>set_persisting ( String name, bool enable )</code>
<code>bool</code>	<code>is_persisting ( String name ) const</code>
<code>void</code>	<code>clear ( String name )</code>
<code>String</code>	<code>localize_path ( String path ) const</code>
<code>String</code>	<code>globalize_path ( String path ) const</code>
<code>int</code>	<code>save ( )</code>
<code>bool</code>	<code>has_singleton ( String name ) const</code>
<code>Object</code>	<code>get_singleton ( String name ) const</code>
<code>bool</code>	<code>load_resource_pack ( String pack )</code>
<code>int</code>	<code>save_custom ( String file )</code>

## 9.97.3 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.

## 9.97.4 Member Function Description

- `bool has ( String name ) const`

Return true if a configuration value is present.

- `void set_order ( String name, int pos )`

Set the order of a configuration value (influences when saved to the config file).

- `int get_order ( String name ) const`

Return 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.

- `bool is_persisting ( String name ) const`

If returns true, this value can be saved to the configuration file. This is useful for editors.

- `void clear ( String name )`

Clear the whole configuration (not recommended, may break things).

- `String localize_path ( String path ) const`

Convert a path to a localized path (res:// path).

- `String globalize_path ( String path ) const`

Convert a localized path (res://) to a full native OS path.

- `int save ( )`

- `bool has_singleton ( String name ) const`

- `Object get_singleton ( String name ) const`
- `bool load_resource_pack ( String pack )`
- `int save_custom ( String file )`

## 9.98 GraphEdit

**Inherits:** `Control < CanvasItem < Node < Object`

**Category:** Core

### 9.98.1 Brief Description

GraphEdit is an area capable of showing various GraphNodes. It manages connection events between them.

### 9.98.2 Member Functions

Error	<code>connect_node ( String from, int from_port, String to, int to_port )</code>
<code>bool</code>	<code>is_node_connected ( String from, int from_port, String to, int to_port )</code>
void	<code>disconnect_node ( String from, int from_port, String to, int to_port )</code>
<code>Array</code>	<code>get_connection_list () const</code>
<code>Vector2</code>	<code>get_scroll_ofs () const</code>
void	<code>set_zoom ( float p_zoom )</code>
<code>float</code>	<code>get_zoom () const</code>
void	<code>set_right_disconnects ( bool enable )</code>
<code>bool</code>	<code>is_right_disconnects_enabled () const</code>

### 9.98.3 Signals

- `delete_nodes_request ()`
- `duplicate_nodes_request ()`
- `popup_request ( Vector2 p_position )`
- `_begin_node_move ()`
- `disconnection_request ( String from, int from_slot, String to, int to_slot )`
- `connection_request ( String from, int from_slot, String to, int to_slot )`
- `_end_node_move ()`

### 9.98.4 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.

## 9.98.5 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.

- *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.

- 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
- void **set\_zoom** (*float* p\_zoom)
- *float* **get\_zoom** () const
- 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.

- *bool* **is\_right\_disconnects\_enabled** () const

Return true if the disconnection of connections is enabled in the visual GraphEdit. False otherwise.

## 9.99 GraphNode

**Inherits:** *Container* < *Control* < *CanvasItem* < *Node* < *Object*

**Category:** Core

### 9.99.1 Brief Description

A GraphNode is a container with several input and output slots allowing connections between GraphNodes. Slots can have different, incompatible types.

## 9.99.2 Member Functions

void	<code>set_title ( String title )</code>
<code>String</code>	<code>get_title () const</code>
void	<code>set_slot ( int idx, bool enable_left, int type_left, Color color_left, bool enable_right, int type_right, Color color_right )</code>
void	<code>clear_slot ( int idx )</code>
void	<code>clear_all_slots ()</code>
<code>bool</code>	<code>is_slot_enabled_left ( int idx ) const</code>
<code>int</code>	<code>get_slot_type_left ( int idx ) const</code>
<code>Color</code>	<code>get_slot_color_left ( int idx ) const</code>
<code>bool</code>	<code>is_slot_enabled_right ( int idx ) const</code>
<code>int</code>	<code>get_slot_type_right ( int idx ) const</code>
<code>Color</code>	<code>get_slot_color_right ( int idx ) const</code>
void	<code>set_offset ( Vector2 offset )</code>
<code>Vector2</code>	<code>get_offset () const</code>
<code>int</code>	<code>get_connection_output_count ()</code>
<code>int</code>	<code>get_connection_input_count ()</code>
<code>Vector2</code>	<code>get_connection_output_pos ( int idx )</code>
<code>int</code>	<code>get_connection_output_type ( int idx )</code>
<code>Color</code>	<code>get_connection_output_color ( int idx )</code>
<code>Vector2</code>	<code>get_connection_input_pos ( int idx )</code>
<code>int</code>	<code>get_connection_input_type ( int idx )</code>
<code>Color</code>	<code>get_connection_input_color ( int idx )</code>
void	<code>set_show_close_button ( bool show )</code>
<code>bool</code>	<code>is_close_button_visible () const</code>

## 9.99.3 Signals

- `raise_request ()`
- `close_request ()`
- `dragged ( Vector2 from, Vector2 to )`
- `offset_changed ()`

## 9.99.4 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.

## 9.99.5 Member Function Description

- void `set_title ( String title )`

Set the title of the GraphNode.

- `String get_title() const`

Return the title of the GraphNode.

- `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.

- `void clear_slot( int idx )`

Disable input and output slot whose index is ‘idx’.

- `void clear_all_slots()`

Disable all input and output slots of the GraphNode.

- `bool is_slot_enabled_left( int idx ) const`

Return true if left (input) slot ‘idx’ is enabled. False otherwise.

- `int get_slot_type_left( int idx ) const`

Return the (integer) type of left (input) ‘idx’ slot.

- `Color get_slot_color_left( int idx ) const`

Return the color set to ‘idx’ left (input) slot.

- `bool is_slot_enabled_right( int idx ) const`

Return true if right (output) slot ‘idx’ is enabled. False otherwise.

- `int get_slot_type_right( int idx ) const`

Return the (integer) type of right (output) ‘idx’ slot.

- `Color get_slot_color_right( int idx ) const`

Return the color set to ‘idx’ right (output) slot.

- `void set_offset( Vector2 offset )`

Set the offset of the GraphNode.

- `Vector2 get_offset() const`

Return the offset of the GraphNode.

- `int get_connection_output_count()`

Return the number of enabled output slots (connections) of the GraphNode.

- `int get_connection_input_count()`

Return the number of enabled input slots (connections) to 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’.

- `Color get_connection_output_color( int idx )`

Return the color of the output connection ‘idx’.

- `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_input_color ( int idx )`

Return the color of the input connection ‘idx’.

- `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.

- `bool is_close_button_visible () const`

Returns true if the close button is shown. False otherwise.

## 9.100 GridContainer

**Inherits:** `Container < Control < CanvasItem < Node < Object`

**Category:** Core

### 9.100.1 Brief Description

### 9.100.2 Member Functions

<code>void</code>	<code>set_columns ( int columns )</code>
<code>int</code>	<code>get_columns () const</code>

### 9.100.3 Member Function Description

- `void set_columns ( int columns )`
- `int get_columns () const`

## 9.101 GridMap

**Inherits:** `Spatial < Node < Object`

**Category:** Core

### 9.101.1 Brief Description

### 9.101.2 Member Functions

<code>void</code>	<code>set_theme ( MeshLibrary theme )</code>
<code>MeshLibrary</code>	<code>get_theme () const</code>
<code>void</code>	<code>set_bake ( bool enable )</code>
Continued on next page	

Table 9.11 – continued from previous page

<i>bool</i>	<i>is_baking_enabled()</i> const
<i>void</i>	<i>set_cell_size( float size )</i>
<i>float</i>	<i>get_cell_size()</i> const
<i>void</i>	<i>set_octant_size( int size )</i>
<i>int</i>	<i>get_octant_size()</i> const
<i>void</i>	<i>set_cell_item( int x, int y, int z, int item, int orientation=0 )</i>
<i>int</i>	<i>get_cell_item( int x, int y, int z )</i> const
<i>int</i>	<i>get_cell_item_orientation( int x, int y, int z )</i> const
<i>void</i>	<i>resource_changed( Object resource )</i>
<i>void</i>	<i>set_center_x( bool enable )</i>
<i>bool</i>	<i>get_center_x()</i> const
<i>void</i>	<i>set_center_y( bool enable )</i>
<i>bool</i>	<i>get_center_y()</i> const
<i>void</i>	<i>set_center_z( bool enable )</i>
<i>bool</i>	<i>get_center_z()</i> const
<i>void</i>	<i>set_clip( bool enabled, bool clipabove=true, int floor=0, int axis=0 )</i>
<i>int</i>	<i>create_area( int id, AABB area )</i>
<i>AABB</i>	<i>area_get_bounds( int area )</i> const
<i>void</i>	<i>area_set_exterior_portal( int area, bool enable )</i>
<i>void</i>	<i>area_set_name( int area, String name )</i>
<i>String</i>	<i>area_get_name( int area )</i> const
<i>bool</i>	<i>area_is_exterior_portal( int area )</i> const
<i>void</i>	<i>area_set_portal_disable_distance( int area, float distance )</i>
<i>float</i>	<i>area_get_portal_disable_distance( int area )</i> const
<i>void</i>	<i>area_set_portal_disable_color( int area, Color color )</i>
<i>Color</i>	<i>area_get_portal_disable_color( int area )</i> const
<i>void</i>	<i>erase_area( int area )</i>
<i>int</i>	<i>get_unused_area_id()</i> const
<i>void</i>	<i>bake_geometry()</i>
<i>void</i>	<i>set_use_baked_light( bool use )</i>
<i>bool</i>	<i>is_using_baked_light()</i> const
<i>void</i>	<i>clear()</i>

### 9.101.3 Numeric Constants

- **INVALID\_CELL\_ITEM = -1**

### 9.101.4 Member Function Description

- *void set\_theme( MeshLibrary theme )*
- *MeshLibrary get\_theme()* const
- *void set\_bake( bool enable )*
- *bool is\_baking\_enabled()* const
- *void set\_cell\_size( float size )*
- *float get\_cell\_size()* const
- *void set\_octant\_size( int size )*
- *int get\_octant\_size()* const

- void **set\_cell\_item** ( *int* x, *int* y, *int* z, *int* item, *int* orientation=0 )
- *int* **get\_cell\_item** ( *int* x, *int* y, *int* z ) const
- *int* **get\_cell\_item\_orientation** ( *int* x, *int* y, *int* z ) const
- void **resource\_changed** ( *Object* resource )
- void **set\_center\_x** ( *bool* enable )
- *bool* **get\_center\_x** ( ) const
- void **set\_center\_y** ( *bool* enable )
- *bool* **get\_center\_y** ( ) const
- void **set\_center\_z** ( *bool* enable )
- *bool* **get\_center\_z** ( ) const
- void **set\_clip** ( *bool* enabled, *bool* clipabove=true, *int* floor=0, *int* axis=0 )
- *int* **create\_area** ( *int* id, *AABB* area )
- *AABB* **area\_get\_bounds** ( *int* area ) const
- void **area\_set\_exterior\_portal** ( *int* area, *bool* enable )
- void **area\_set\_name** ( *int* area, *String* name )
- *String* **area\_get\_name** ( *int* area ) const
- *bool* **area\_is\_exterior\_portal** ( *int* area ) const
- void **area\_set\_portal\_disable\_distance** ( *int* area, *float* distance )
- *float* **area\_get\_portal\_disable\_distance** ( *int* area ) const
- void **area\_set\_portal\_disable\_color** ( *int* area, *Color* color )
- *Color* **area\_get\_portal\_disable\_color** ( *int* area ) const
- void **erase\_area** ( *int* area )
- *int* **get\_unused\_area\_id** ( ) const
- void **bake\_geometry** ( )
- void **set\_use\_baked\_light** ( *bool* use )
- *bool* **is\_using\_baked\_light** ( ) const
- void **clear** ( )

## 9.102 GrooveJoint2D

Inherits: *Joint2D* < *Node2D* < *CanvasItem* < *Node* < *Object*

Category: Core

### 9.102.1 Brief Description

Groove constraint for 2D physics.

## 9.102.2 Member Functions

void	<code>set_length (float length)</code>
<code>float</code>	<code>get_length () const</code>
void	<code>set_initial_offset (float offset)</code>
<code>float</code>	<code>get_initial_offset () const</code>

## 9.102.3 Description

Groove constraint for 2D physics. This is useful for making a body “slide” through a segment placed in another.

## 9.102.4 Member Function Description

- void `set_length (float length)`

Set the length of the groove.

- `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.

- `float get_initial_offset () const`

Set the final offset of the groove on body A.

## 9.103 HBoxContainer

**Inherits:** `BoxContainer < Container < Control < CanvasItem < Node < Object`

**Category:** Core

### 9.103.1 Brief Description

Horizontal box container.

### 9.103.2 Description

Horizontal box container. See `BoxContainer`.

## 9.104 HButtonArray

**Inherits:** `ButtonArray < Control < CanvasItem < Node < Object`

**Category:** Core

### 9.104.1 Brief Description

Horizontal button array.

### 9.104.2 Description

Horizontal button array. See [ButtonArray](#).

## 9.105 HingeJoint

**Inherits:** [Joint](#) < [Spatial](#) < [Node](#) < [Object](#)

**Category:** Core

### 9.105.1 Brief Description

### 9.105.2 Member Functions

void	<code>set_param ( int param, float value )</code>
<code>float</code>	<code>get_param ( int param ) const</code>
void	<code>set_flag ( int flag, bool enabled )</code>
<code>bool</code>	<code>get_flag ( int flag ) const</code>

### 9.105.3 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**

## 9.105.4 Member Function Description

- void `set_param` ( `int` param, `float` value )
- `float get_param` ( `int` param ) const
- void `set_flag` ( `int` flag, `bool` enabled )
- `bool get_flag` ( `int` flag ) const

## 9.106 HScrollBar

**Inherits:** `ScrollBar < Range < Control < CanvasItem < Node < Object`

**Category:** Core

### 9.106.1 Brief Description

Horizontal scroll bar.

### 9.106.2 Description

Horizontal scroll bar. See `ScrollBar`. This one goes from left (min) to right (max).

## 9.107 HSeparator

**Inherits:** `Separator < Control < CanvasItem < Node < Object`

**Category:** Core

### 9.107.1 Brief Description

Horizontal separator.

### 9.107.2 Description

Horizontal separator. See `Separator`. It is used to separate objects vertically, though (but it looks horizontal!).

## 9.108 HSlider

**Inherits:** `Slider < Range < Control < CanvasItem < Node < Object`

**Category:** Core

### 9.108.1 Brief Description

Horizontal slider.

## 9.108.2 Description

Horizontal slider. See [Slider](#). This one goes from left (min) to right (max).

## 9.109 HSplitContainer

**Inherits:** [SplitContainer](#) < [Container](#) < [Control](#) < [CanvasItem](#) < [Node](#) < [Object](#)

**Category:** Core

### 9.109.1 Brief Description

Horizontal split container.

### 9.109.2 Description

Horizontal split container. See [SplitContainer](#). This goes from left to right.

## 9.110 HTTPClient

**Inherits:** [Reference](#) < [Object](#)

**Category:** Core

### 9.110.1 Brief Description

### 9.110.2 Member Functions

Error	<code>connect ( String host, int port, bool use_ssl=false, bool verify_host=true )</code>
void	<code>set_connection ( StreamPeer connection )</code>
int	<code>request ( int method, String url, StringArray headers, String body="" )</code>
int	<code>send_body_text ( String body )</code>
int	<code>send_body_data ( RawArray body )</code>
void	<code>close ()</code>
bool	<code>has_response () const</code>
bool	<code>is_response_chunked () const</code>
int	<code>get_response_code () const</code>
StringArray	<code>get_response_headers ()</code>
Dictionary	<code>get_response_headers_as_dictionary ()</code>
int	<code>get_response_body_length () const</code>
RawArray	<code>read_response_body_chunk ()</code>
void	<code>set_read_chunk_size ( int bytes )</code>
void	<code>set_blocking_mode ( bool enabled )</code>
bool	<code>is_blocking_mode_enabled () const</code>
int	<code>get_status () const</code>
Error	<code>poll ()</code>
String	<code>query_string_from_dict ( Dictionary fields )</code>

### 9.110.3 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**
- **RESPONSE\_MOVED\_PERMANENTLY = 301**
- **RESPONSE\_FOUND = 302**
- **RESPONSE\_SEE\_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

## 9.110.4 Member Function Description

- 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.

- void **set\_connection** ( *StreamPeer* connection )
- *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 queryString = httpClient.query_string_from_dict(fields)  
  
var headers = :ref:`Content-Type: application/x-www-form-urlencoded`, "Content-Length: " + str(queryString)  
  
var result = httpClient.request(httpClient.METHOD_POST, "index.php", headers, queryString)
```

- *int* **send\_body\_text** ( *String* body )

Stub function

- *int* **send\_body\_data** ( *RawArray* body )

Stub function

- void **close** ()
- *bool* **has\_response** () const
- *bool* **is\_response\_chunked** () const
- *int* **get\_response\_code** () const
- *StringArray* **get\_response\_headers** ()
- *Dictionary* **get\_response\_headers\_as\_dictionary** ()
- *int* **get\_response\_body\_length** () const
- *RawArray* **read\_response\_body\_chunk** ()
- void **set\_read\_chunk\_size** ( *int* bytes )

Sets the size of the buffer used and maximum bytes to read per iteration

- void **set\_blocking\_mode** ( *bool* enabled )

If set to true, execute will wait until all data is read from the response.

- *bool* **is\_blocking\_mode\_enabled** () const
- *int* **get\_status** () const

Returns a status string like STATUS\_REQUESTING. Need to call `poll` in order to get status updates.

- 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"
```

## 9.111 Image

**Category:** Built-In Types

### 9.111.1 Brief Description

Image datatype.

### 9.111.2 Member Functions

void	<code>blit_rect ( Image src, Rect2 src_rect, Vector2 dest=0 )</code>
void	<code>brush_transfer ( Image src, Image brush, Vector2 pos=0 )</code>
<code>Image</code>	<code>brushed ( Image src, Image brush, Vector2 pos=0 )</code>
<code>Image</code>	<code>compressed ( int format=0 )</code>
<code>Image</code>	<code>converted ( int format=0 )</code>
<code>Image</code>	<code>decompressed ()</code>
<code>bool</code>	<code>empty ()</code>
<code>RawArray</code>	<code>get_data ()</code>
<code>int</code>	<code>get_format ()</code>
<code>int</code>	<code>get_height ()</code>
<code>Color</code>	<code>get_pixel ( int x, int y, int mipmap_level=0 )</code>
<code>Image</code>	<code>get_rect ( Rect2 area=0 )</code>
<code>Rect2</code>	<code>get_used_rect ()</code>
<code>int</code>	<code>get_width ()</code>
<code>int</code>	<code>load ( String path=0 )</code>
void	<code>put_pixel ( int x, int y, Color color, int mipmap_level=0 )</code>
<code>Image</code>	<code>resized ( int x, int y, int interpolation=1 )</code>
<code>int</code>	<code>save_png ( String path=0 )</code>
<code>Image</code>	<code>Image ( int width, int height, bool mipmaps, int format )</code>

### 9.111.3 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

#### 9.111.4 Description

Built in native image datatype. Contains image data, which can be converted to a texture, and several functions to interact with it.

#### 9.111.5 Member Function Description

- void **blit\_rect** (*Image* src, *Rect2* src\_rect, *Vector2* dest=0 )
- void **brush\_transfer** (*Image* src, *Image* brush, *Vector2* pos=0 )
- *Image* **brushed** (*Image* src, *Image* brush, *Vector2* pos=0 )
- *Image* **compressed** (*int* format=0 )
- *Image* **converted** (*int* format=0 )
- *Image* **decompressed** ()
- *bool* **empty** ()

- `RawArray get_data()`
- `int get_format()`
- `int get_height()`
- `Color get_pixel( int x, int y, int mipmap_level=0 )`
- `Image get_rect( Rect2 area=0 )`
- `Rect2 get_used_rect()`
- `int get_width()`
- `int load( String path=0 )`
- `void put_pixel( int x, int y, Color color, int mipmap_level=0 )`
- `Image resized( int x, int y, int interpolation=1 )`
- `int save_png( String path=0 )`
- `Image Image( int width, int height, bool mipmaps, int format )`

Create an empty image of a specific size and format.

## 9.112 ImageTexture

**Inherits:** `Texture < Resource < Reference < Object`

**Category:** Core

### 9.112.1 Brief Description

### 9.112.2 Member Functions

<code>void</code>	<code>create( int width, int height, int format, int flags=7 )</code>
<code>void</code>	<code>create_from_image( Image image, int flags=7 )</code>
<code>int</code>	<code>get_format() const</code>
<code>void</code>	<code>load( String path )</code>
<code>void</code>	<code>set_data( Image image )</code>
<code>Image</code>	<code>get_data() const</code>
<code>void</code>	<code>set_storage( int mode )</code>
<code>int</code>	<code>get_storage() const</code>
<code>void</code>	<code>set_lossy_storage_quality( float quality )</code>
<code>float</code>	<code>get_lossy_storage_quality() const</code>
<code>void</code>	<code>fix_alpha_edges()</code>
<code>void</code>	<code>premultiply_alpha()</code>
<code>void</code>	<code>normal_to_xy()</code>
<code>void</code>	<code>shrink_x2_and_keep_size()</code>
<code>void</code>	<code>set_size_override( Vector2 size )</code>

### 9.112.3 Numeric Constants

- `STORAGE_RAW = 0`
- `STORAGE_COMPRESS_LOSSY = 1`

- STORAGE\_COMPRESS\_LOSSLESS = 2

#### 9.112.4 Member Function Description

- void **create** ( *int* width, *int* height, *int* format, *int* flags=7 )
- void **create\_from\_image** ( *Image* image, *int* flags=7 )
- *int* **get\_format** ( ) const
- void **load** ( *String* path )
- void **set\_data** ( *Image* image )
- *Image* **get\_data** ( ) const
- void **set\_storage** ( *int* mode )
- *int* **get\_storage** ( ) const
- void **set\_lossy\_storage\_quality** ( *float* quality )
- *float* **get\_lossy\_storage\_quality** ( ) const
- void **fix\_alpha\_edges** ( )
- void **premultiply\_alpha** ( )
- void **normal\_to\_xy** ( )
- void **shrink\_x2\_and\_keep\_size** ( )
- void **set\_size\_override** ( *Vector2* size )

### 9.113 ImmediateGeometry

**Inherits:** *GeometryInstance* < *VisualInstance* < *Spatial* < *Node* < *Object*

**Category:** Core

#### 9.113.1 Brief Description

#### 9.113.2 Member Functions

void	<i>begin</i> ( <i>int</i> primitive, <i>Texture</i> texture )
void	<i>set_normal</i> ( <i>Vector3</i> normal )
void	<i>set_tangent</i> ( <i>Plane</i> tangent )
void	<i>set_color</i> ( <i>Color</i> color )
void	<i>set_uv</i> ( <i>Vector2</i> uv )
void	<i>set_uv2</i> ( <i>Vector2</i> uv )
void	<i>add_vertex</i> ( <i>Vector3</i> pos )
void	<i>add_sphere</i> ( <i>int</i> lats, <i>int</i> lons, <i>float</i> radius )
void	<i>end</i> ( )
void	<i>clear</i> ( )

### 9.113.3 Member Function Description

- void **begin** ( *int* primitive, *Texture* texture )
- void **set\_normal** ( *Vector3* normal )
- void **set\_tangent** ( *Plane* tangent )
- void **set\_color** ( *Color* color )
- void **set\_uv** ( *Vector2* uv )
- void **set\_uv2** ( *Vector2* uv )
- void **add\_vertex** ( *Vector3* pos )
- void **add\_sphere** ( *int* lats, *int* lons, *float* radius )
- void **end** ( )
- void **clear** ( )

## 9.114 Input

Inherits: *Object*

Inherited By: *InputDefault*

Category: Core

### 9.114.1 Brief Description

### 9.114.2 Member Functions

<i>bool</i>	<i>is_key_pressed</i> ( <i>int</i> scancode )
<i>bool</i>	<i>is_mouse_button_pressed</i> ( <i>int</i> button )
<i>bool</i>	<i>is_joy_button_pressed</i> ( <i>int</i> device, <i>int</i> button )
<i>bool</i>	<i>is_action_pressed</i> ( <i>String</i> action )
<i>void</i>	<i>add_joy_mapping</i> ( <i>String</i> mapping, <i>bool</i> update_existing=false )
<i>void</i>	<i>remove_joy_mapping</i> ( <i>String</i> guid )
<i>bool</i>	<i>is_joy_known</i> ( <i>int</i> device )
<i>float</i>	<i>get_joy_axis</i> ( <i>int</i> device, <i>int</i> axis )
<i>String</i>	<i>get_joy_name</i> ( <i>int</i> device )
<i>String</i>	<i>get_joy_guid</i> ( <i>int</i> device ) const
<i>Vector3</i>	<i>get_accelerometer</i> ( )
<i>Vector2</i>	<i>get_mouse_speed</i> ( ) const
<i>int</i>	<i>get_mouse_button_mask</i> ( ) const
<i>void</i>	<i>set_mouse_mode</i> ( <i>int</i> mode )
<i>int</i>	<i>get_mouse_mode</i> ( ) const
<i>void</i>	<i>warp_mouse_pos</i> ( <i>Vector2</i> to )
<i>void</i>	<i>action_press</i> ( <i>String</i> action )
<i>void</i>	<i>action_release</i> ( <i>String</i> action )
<i>void</i>	<i>set_custom_mouse_cursor</i> ( <i>Texture</i> image, <i>Vector2</i> hotspot=Vector2(0,0) )

### 9.114.3 Signals

- **joy\_connection\_changed** ( *int* index, *bool* connected )

### 9.114.4 Numeric Constants

- **MOUSE\_MODE\_VISIBLE** = 0
- **MOUSE\_MODE\_HIDDEN** = 1
- **MOUSE\_MODE\_CAPTURED** = 2

### 9.114.5 Member Function Description

- *bool* **is\_key\_pressed** ( *int* scancode )
- *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](#).

- *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\_action\_pressed** ( *String* action )
- *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.

- *void* **remove\_joy\_mapping** ( *String* guid )

Removes all mappings from the internal db that match the given uid.

- *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.

- *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](#))

- *String* **get\_joy\_name** ( *int* device )

Returns the name of the joystick at the specified device index

- *String* **get\_joy\_guid** ( *int* device ) const

Returns a SDL2 compatible device guid on platforms that use gamepad remapping. Returns “Default Gamepad” otherwise.

- *Vector3* **get\_accelerometer** ()
- *Vector2* **get\_mouse\_speed** () const
- *int* **get\_mouse\_button\_mask** () const
- *void* **set\_mouse\_mode** ( *int* mode )
- *int* **get\_mouse\_mode** () const
- *void* **warp\_mouse\_pos** ( *Vector2* to )

- void **action\_press** ( *String* action )
- void **action\_release** ( *String* action )
- void **set\_custom\_mouse\_cursor** ( *Texture* image, *Vector2* hotspot=Vector2(0,0) )

## 9.115 InputDefault

Inherits: *Input* < *Object*

Category: Core

### 9.115.1 Brief Description

## 9.116 InputEvent

Category: Built-In Types

### 9.116.1 Brief Description

Built-in input event data.

### 9.116.2 Member Functions

<i>bool</i>	<i>is_action</i> ( <i>String</i> action )
<i>bool</i>	<i>is_action_pressed</i> ( <i>String</i> is_action_pressed )
<i>bool</i>	<i>is_action_released</i> ( <i>String</i> is_action_released )
<i>bool</i>	<i>is_echo</i> ()
<i>bool</i>	<i>is_pressed</i> ()
<i>void</i>	<i>set_as_action</i> ( <i>String</i> action, <i>bool</i> pressed )

### 9.116.3 Member Variables

- *int* **type**
- *int* **device**
- *int* **ID**

### 9.116.4 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\_DRAG** = 7
- **ACTION** = 8

## 9.116.5 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).

## 9.116.6 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* is\_action\_pressed)
- *bool* **is\_action\_released** (*String* is\_action\_released)
- *bool* **is\_echo** ()

Return if this input event is an echo event (usually for key events).

- *bool* **is\_pressed** ()

Return if this input event is pressed (for key, mouse, joy button or screen press events).

- *void* **set\_as\_action** (*String* action, *bool* pressed)

## 9.117 InputEventAction

**Category:** Built-In Types

### 9.117.1 Brief Description

### 9.117.2 Member Functions

<i>bool</i>	<b>is_action</b> ( <i>String</i> action)
<i>bool</i>	<b>is_action_pressed</b> ( <i>String</i> is_action_pressed)
<i>bool</i>	<b>is_action_released</b> ( <i>String</i> is_action_released)
<i>bool</i>	<b>is_echo</b> ()
<i>bool</i>	<b>is_pressed</b> ()
<i>void</i>	<b>set_as_action</b> ( <i>String</i> action, <i>bool</i> pressed)

### 9.117.3 Member Variables

- *int* **type**
- *int* **device**
- *int* **ID**

## 9.117.4 Numeric Constants

- **NONE = 0**
- **KEY = 1**
- **MOUSE\_MOTION = 2**
- **MOUSE\_BUTTON = 3**
- **JOYSTICK\_MOTION = 4**
- **JOYSTICK\_BUTTON = 5**
- **SCREEN\_TOUCH = 6**
- **SCREEN\_DRAG = 7**
- **ACTION = 8**

## 9.117.5 Member Function Description

- *bool* **is\_action** ( *String* action )
- *bool* **is\_action\_pressed** ( *String* is\_action\_pressed )
- *bool* **is\_action\_released** ( *String* is\_action\_released )
- *bool* **is\_echo** ( )
- *bool* **is\_pressed** ( )
- *void* **set\_as\_action** ( *String* action, *bool* pressed )

## 9.118 InputEventJoyButton

**Category:** Built-In Types

### 9.118.1 Brief Description

### 9.118.2 Member Functions

<i>bool</i>	<b>is_action</b> ( <i>String</i> action )
<i>bool</i>	<b>is_action_pressed</b> ( <i>String</i> is_action_pressed )
<i>bool</i>	<b>is_action_released</b> ( <i>String</i> is_action_released )
<i>bool</i>	<b>is_echo</b> ( )
<i>bool</i>	<b>is_pressed</b> ( )
<i>void</i>	<b>set_as_action</b> ( <i>String</i> action, <i>bool</i> pressed )

### 9.118.3 Member Variables

- *int* **type**
- *int* **device**
- *int* **ID**

- *int* **button\_index**
- *bool* **pressed**
- *float* **pressure**

## 9.118.4 Numeric Constants

- **NONE = 0**
- **KEY = 1**
- **MOUSE\_MOTION = 2**
- **MOUSE\_BUTTON = 3**
- **JOYSTICK\_MOTION = 4**
- **JOYSTICK\_BUTTON = 5**
- **SCREEN\_TOUCH = 6**
- **SCREEN\_DRAG = 7**
- **ACTION = 8**

## 9.118.5 Member Function Description

- *bool* **is\_action** ( *String* action )
- *bool* **is\_action\_pressed** ( *String* is\_action\_pressed )
- *bool* **is\_action\_released** ( *String* is\_action\_released )
- *bool* **is\_echo** ()
- *bool* **is\_pressed** ()
- *void* **set\_as\_action** ( *String* action, *bool* pressed )

## 9.119 InputEventJoyMotion

**Category:** Built-In Types

### 9.119.1 Brief Description

### 9.119.2 Member Functions

<i>bool</i>	<i>is_action</i> ( <i>String</i> action )
<i>bool</i>	<i>is_action_pressed</i> ( <i>String</i> is_action_pressed )
<i>bool</i>	<i>is_action_released</i> ( <i>String</i> is_action_released )
<i>bool</i>	<i>is_echo</i> ()
<i>bool</i>	<i>is_pressed</i> ()
<i>void</i>	<i>set_as_action</i> ( <i>String</i> action, <i>bool</i> pressed )

### 9.119.3 Member Variables

- *int* **type**
- *int* **device**
- *int* **ID**
- *int* **axis**
- *float* **value**

### 9.119.4 Numeric Constants

- **NONE = 0**
- **KEY = 1**
- **MOUSE\_MOTION = 2**
- **MOUSE\_BUTTON = 3**
- **JOYSTICK\_MOTION = 4**
- **JOYSTICK\_BUTTON = 5**
- **SCREEN\_TOUCH = 6**
- **SCREEN\_DRAG = 7**
- **ACTION = 8**

### 9.119.5 Member Function Description

- *bool* **is\_action** ( *String* action )
- *bool* **is\_action\_pressed** ( *String* is\_action\_pressed )
- *bool* **is\_action\_released** ( *String* is\_action\_released )
- *bool* **is\_echo** ( )
- *bool* **is\_pressed** ( )
- *void* **set\_as\_action** ( *String* action, *bool* pressed )

## 9.120 InputEventKey

**Category:** Built-In Types

## 9.120.1 Brief Description

## 9.120.2 Member Functions

<code>bool</code>	<code>is_action ( String action )</code>
<code>bool</code>	<code>is_action_pressed ( String is_action_pressed )</code>
<code>bool</code>	<code>is_action_released ( String is_action_released )</code>
<code>bool</code>	<code>is_echo ()</code>
<code>bool</code>	<code>is_pressed ()</code>
<code>void</code>	<code>set_as_action ( String action, bool pressed )</code>

## 9.120.3 Member Variables

- `int type`
- `int device`
- `int ID`
- `bool shift`
- `bool alt`
- `bool control`
- `bool meta`
- `bool pressed`
- `bool echo`
- `int scancode`
- `int unicode`

## 9.120.4 Numeric Constants

- `NONE = 0`
- `KEY = 1`
- `MOUSE_MOTION = 2`
- `MOUSE_BUTTON = 3`
- `JOYSTICK_MOTION = 4`
- `JOYSTICK_BUTTON = 5`
- `SCREEN_TOUCH = 6`
- `SCREEN_DRAG = 7`
- `ACTION = 8`

## 9.120.5 Member Function Description

- `bool is_action ( String action )`
- `bool is_action_pressed ( String is_action_pressed )`

- `bool is_action_released ( String is_action_released )`
- `bool is_echo ()`
- `bool is_pressed ()`
- `void set_as_action ( String action, bool pressed )`

## 9.121 InputEventMouseButton

**Category:** Built-In Types

### 9.121.1 Brief Description

### 9.121.2 Member Functions

<code>bool</code>	<code>is_action ( String action )</code>
<code>bool</code>	<code>is_action_pressed ( String is_action_pressed )</code>
<code>bool</code>	<code>is_action_released ( String is_action_released )</code>
<code>bool</code>	<code>is_echo ()</code>
<code>bool</code>	<code>is_pressed ()</code>
<code>void</code>	<code>set_as_action ( String action, bool pressed )</code>

### 9.121.3 Member Variables

- `int type`
- `int device`
- `int ID`
- `bool shift`
- `bool alt`
- `bool control`
- `bool meta`
- `int button_mask`
- `int x`
- `int y`
- `Vector2 pos`
- `int global_x`
- `int global_y`
- `Vector2 global_pos`
- `int button_index`
- `bool pressed`
- `bool doubleclick`

## 9.121.4 Numeric Constants

- **NONE** = 0
- **KEY** = 1
- **MOUSE\_MOTION** = 2
- **MOUSE\_BUTTON** = 3
- **JOYSTICK\_MOTION** = 4
- **JOYSTICK\_BUTTON** = 5
- **SCREEN\_TOUCH** = 6
- **SCREEN\_DRAG** = 7
- **ACTION** = 8

## 9.121.5 Member Function Description

- *bool* **is\_action** ( *String* action )
- *bool* **is\_action\_pressed** ( *String* is\_action\_pressed )
- *bool* **is\_action\_released** ( *String* is\_action\_released )
- *bool* **is\_echo** ( )
- *bool* **is\_pressed** ( )
- *void* **set\_as\_action** ( *String* action, *bool* pressed )

## 9.122 InputEventMouseMotion

**Category:** Built-In Types

### 9.122.1 Brief Description

### 9.122.2 Member Functions

<i>bool</i>	<b>is_action</b> ( <i>String</i> action )
<i>bool</i>	<b>is_action_pressed</b> ( <i>String</i> is_action_pressed )
<i>bool</i>	<b>is_action_released</b> ( <i>String</i> is_action_released )
<i>bool</i>	<b>is_echo</b> ( )
<i>bool</i>	<b>is_pressed</b> ( )
<i>void</i>	<b>set_as_action</b> ( <i>String</i> action, <i>bool</i> pressed )

### 9.122.3 Member Variables

- *int* **type**
- *int* **device**
- *int* **ID**

- *bool* shift
- *bool* alt
- *bool* control
- *bool* meta
- *int* button\_mask
- *int* x
- *int* y
- *Vector2* pos
- *int* global\_x
- *int* global\_y
- *Vector2* global\_pos
- *int* relative\_x
- *int* relative\_y
- *Vector2* relative\_pos
- *float* speed\_x
- *float* speed\_y
- *Vector2* speed

## 9.122.4 Numeric Constants

- **NONE = 0**
- **KEY = 1**
- **MOUSE\_MOTION = 2**
- **MOUSE\_BUTTON = 3**
- **JOYSTICK\_MOTION = 4**
- **JOYSTICK\_BUTTON = 5**
- **SCREEN\_TOUCH = 6**
- **SCREEN\_DRAG = 7**
- **ACTION = 8**

## 9.122.5 Member Function Description

- *bool* is\_action ( *String* action )
- *bool* is\_action\_pressed ( *String* is\_action\_pressed )
- *bool* is\_action\_released ( *String* is\_action\_released )
- *bool* is\_echo ()
- *bool* is\_pressed ()
- void set\_as\_action ( *String* action, *bool* pressed )

## 9.123 InputEventScreenDrag

**Category:** Built-In Types

### 9.123.1 Brief Description

### 9.123.2 Member Functions

<code>bool</code>	<code>is_action ( String action )</code>
<code>bool</code>	<code>is_action_pressed ( String is_action_pressed )</code>
<code>bool</code>	<code>is_action_released ( String is_action_released )</code>
<code>bool</code>	<code>is_echo ()</code>
<code>bool</code>	<code>is_pressed ()</code>
<code>void</code>	<code>set_as_action ( String action, bool pressed )</code>

### 9.123.3 Member Variables

- `int type`
- `int device`
- `int ID`
- `int index`
- `float x`
- `float y`
- `Vector2 pos`
- `float relative_x`
- `float relative_y`
- `Vector2 relative_pos`
- `float speed_x`
- `float speed_y`
- `Vector2 speed`

### 9.123.4 Numeric Constants

- `NONE = 0`
- `KEY = 1`
- `MOUSE_MOTION = 2`
- `MOUSE_BUTTON = 3`
- `JOYSTICK_MOTION = 4`
- `JOYSTICK_BUTTON = 5`
- `SCREEN_TOUCH = 6`
- `SCREEN_DRAG = 7`

- ACTION = 8

### 9.123.5 Member Function Description

- *bool* `is_action ( String action )`
- *bool* `is_action_pressed ( String is_action_pressed )`
- *bool* `is_action_released ( String is_action_released )`
- *bool* `is_echo ()`
- *bool* `is_pressed ()`
- `void set_as_action ( String action, bool pressed )`

## 9.124 InputEventScreenTouch

**Category:** Built-In Types

### 9.124.1 Brief Description

### 9.124.2 Member Functions

<i>bool</i>	<code>is_action ( String action )</code>
<i>bool</i>	<code>is_action_pressed ( String is_action_pressed )</code>
<i>bool</i>	<code>is_action_released ( String is_action_released )</code>
<i>bool</i>	<code>is_echo ()</code>
<i>bool</i>	<code>is_pressed ()</code>
<code>void</code>	<code>set_as_action ( String action, bool pressed )</code>

### 9.124.3 Member Variables

- *int* `type`
- *int* `device`
- *int* `ID`
- *int* `index`
- *float* `x`
- *float* `y`
- `Vector2 pos`
- *bool* `pressed`

### 9.124.4 Numeric Constants

- `NONE = 0`
- `KEY = 1`

- **MOUSE\_MOTION** = 2
- **MOUSE\_BUTTON** = 3
- **JOYSTICK\_MOTION** = 4
- **JOYSTICK\_BUTTON** = 5
- **SCREEN\_TOUCH** = 6
- **SCREEN\_DRAG** = 7
- **ACTION** = 8

## 9.124.5 Member Function Description

- *bool* **is\_action** ( *String* action )
- *bool* **is\_action\_pressed** ( *String* is\_action\_pressed )
- *bool* **is\_action\_released** ( *String* is\_action\_released )
- *bool* **is\_echo** ( )
- *bool* **is\_pressed** ( )
- void **set\_as\_action** ( *String* action, *bool* pressed )

## 9.125 InputMap

**Inherits:** *Object*

**Category:** Core

### 9.125.1 Brief Description

Singleton that manages actions.

### 9.125.2 Member Functions

<i>bool</i>	<b>has_action</b> ( <i>String</i> action ) const
<i>int</i>	<b>get_action_id</b> ( <i>String</i> action ) const
<i>String</i>	<b>get_action_from_id</b> ( <i>int</i> id ) const
<i>void</i>	<b>add_action</b> ( <i>String</i> action )
<i>void</i>	<b>erase_action</b> ( <i>String</i> action )
<i>void</i>	<b>action_add_event</b> ( <i>String</i> action, <i>InputEvent</i> event )
<i>bool</i>	<b>action_has_event</b> ( <i>String</i> action, <i>InputEvent</i> event )
<i>void</i>	<b>action_erase_event</b> ( <i>String</i> action, <i>InputEvent</i> event )
<i>Array</i>	<b>get_action_list</b> ( <i>String</i> action )
<i>bool</i>	<b>event_is_action</b> ( <i>InputEvent</i> event, <i>String</i> action ) const
<i>void</i>	<b>load_from_globals</b> ( )

### 9.125.3 Description

Singleton that manages actions. InputMap has a list of the actions used in InputEvent, which can be modified.

### 9.125.4 Member Function Description

- `bool has_action ( String action ) const`
- `int get_action_id ( String action ) const`
- `String get_action_from_id ( int id ) const`
- `void add_action ( String action )`
- `void erase_action ( String action )`
- `void action_add_event ( String action, InputEvent event )`
- `bool action_has_event ( String action, InputEvent event )`
- `void action_erase_event ( String action, InputEvent event )`
- `Array get_action_list ( String action )`
- `bool event_is_action ( InputEvent event, String action ) const`
- `void load_from_globals ()`

## 9.126 InstancePlaceholder

**Inherits:** `Node < Object`

**Category:** Core

### 9.126.1 Brief Description

### 9.126.2 Member Functions

<code>void</code>	<code>replace_by_instance ( PackedScene custom_scene=NULL )</code>
<code>String</code>	<code>get_instance_path () const</code>

### 9.126.3 Member Function Description

- `void replace_by_instance ( PackedScene custom_scene=NULL )`
- `String get_instance_path () const`

## 9.127 int

**Category:** Built-In Types

### 9.127.1 Brief Description

Integer built-in type.

### 9.127.2 Member Functions

<code>int</code>	<code>int ( bool from )</code>
<code>int</code>	<code>int ( float from )</code>
<code>int</code>	<code>int ( String from )</code>

### 9.127.3 Description

Integer built-in type.

### 9.127.4 Member Function Description

- `int int ( bool from )`
- `int int ( float from )`
- `int int ( String from )`

## 9.128 IntArray

**Category:** Built-In Types

### 9.128.1 Brief Description

Integer Array.

### 9.128.2 Member Functions

<code>void</code>	<code>push_back ( int integer )</code>
<code>void</code>	<code>resize ( int idx )</code>
<code>void</code>	<code>set ( int idx, int integer )</code>
<code>int</code>	<code>size ()</code>
<code>IntArray</code>	<code>IntArray ( Array from )</code>

### 9.128.3 Description

Integer Array. Array of integers. Can only contain integers. Optimized for memory usage, can't fragment the memory.

## 9.128.4 Member Function Description

- void **push\_back** ( *int* integer )

Append a value to the array.

- void **resize** ( *int* idx )

Resize the array.

- void **set** ( *int* idx, *int* integer )

Set an index in the array.

- *int* **size** ( )

Return the array size.

- *IntArray* **IntArray** ( *Array* from )

Create from a generic array.

## 9.129 InterpolatedCamera

**Inherits:** *Camera* < *Spatial* < *Node* < *Object*

**Category:** Core

### 9.129.1 Brief Description

### 9.129.2 Member Functions

void	<i>set_target_path</i> ( <i>NodePath</i> target_path )
<i>NodePath</i>	<i>get_target_path</i> ( ) const
void	<i>set_target</i> ( <i>Camera</i> target )
void	<i>set_speed</i> ( <i>float</i> speed )
<i>float</i>	<i>get_speed</i> ( ) const
void	<i>set_interpolation_enabled</i> ( <i>bool</i> target_path )
<i>bool</i>	<i>is_interpolation_enabled</i> ( ) const

### 9.129.3 Member Function Description

- void **set\_target\_path** ( *NodePath* target\_path )
- *NodePath* **get\_target\_path** ( ) const
- void **set\_target** ( *Camera* target )
- void **set\_speed** ( *float* speed )
- *float* **get\_speed** ( ) const
- void **set\_interpolation\_enabled** ( *bool* target\_path )
- *bool* **is\_interpolation\_enabled** ( ) const

## 9.130 IP

**Inherits:** *Object*

**Inherited By:** *IP\_Unix*

**Category:** Core

### 9.130.1 Brief Description

IP Protocol support functions.

### 9.130.2 Member Functions

<i>String</i>	<code>resolve_hostname ( String host )</code>
<i>int</i>	<code>resolve_hostname_queue_item ( String host )</code>
<i>int</i>	<code>get_resolve_item_status ( int id ) const</code>
<i>String</i>	<code>get_resolve_item_address ( int id ) const</code>
<i>void</i>	<code>erase_resolve_item ( int id )</code>
<i>Array</i>	<code>get_local_addresses ( ) const</code>

### 9.130.3 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**

### 9.130.4 Description

IP contains some support functions for the IPv4 protocol. TCP/IP support is in different classes (see *TCP\_Client*, *TCP\_Server*). IP provides hostname resolution support, both blocking and threaded.

### 9.130.5 Member Function Description

- `String resolve_hostname ( String host )`

Resolve a given hostname, blocking. Resolved hostname is returned as an IP.

- `int resolve_hostname_queue_item ( String host )`

Create a queue item for resolving a given hostname. The queue ID is returned, or RESOLVER\_INVALID\_ID on error.

- `int get_resolve_item_status ( int id ) const`

Return the status of hostname queued for resolving, given its queue ID. Returned status can be any of the RESOLVER\_STATUS\_\* enumeration.

- `String get_resolve_item_address ( int id ) const`

Return a resolved item address, or an empty string if an error happened or resolution didn't happen yet (see `get_resolve_item_status`).

- `void erase_resolve_item ( int id )`

Erase a queue ID, removing it from the queue if needed. This should be used after a queue is completed to free it and enable more queries to happen.

- `Array get_local_addresses ( ) const`

## 9.131 IP\_Unix

**Inherits:** `IP < Object`

**Category:** Core

### 9.131.1 Brief Description

## 9.132 ItemList

**Inherits:** `Control < CanvasItem < Node < Object`

**Category:** Core

### 9.132.1 Brief Description

### 9.132.2 Member Functions

<code>void</code>	<code>add_item ( String text, Texture icon=Object(), bool selectable=true )</code>
<code>void</code>	<code>add_icon_item ( Texture icon, bool selectable=true )</code>
<code>void</code>	<code>set_item_text ( int idx, String text )</code>
<code>String</code>	<code>get_item_text ( int idx ) const</code>
<code>void</code>	<code>set_item_icon ( int idx, Texture icon )</code>
<code>Texture</code>	<code>get_item_icon ( int idx ) const</code>
<code>void</code>	<code>set_item_selectable ( int idx, bool selectable )</code>
<code>bool</code>	<code>is_item_selectable ( int idx ) const</code>
<code>void</code>	<code>set_item_disabled ( int idx, bool disabled )</code>
<code>bool</code>	<code>is_item_disabled ( int idx ) const</code>
<code>void</code>	<code>set_item_metadata ( int idx, var metadata )</code>
<code>void</code>	<code>get_item_metadata ( int idx ) const</code>
<code>void</code>	<code>set_item_custom_bg_color ( int idx, Color custom_bg_color )</code>
<code>Color</code>	<code>get_item_custom_bg_color ( int idx ) const</code>
<code>void</code>	<code>set_item_tooltip ( int idx, String tooltip )</code>
<code>String</code>	<code>get_item_tooltip ( int idx ) const</code>
<code>void</code>	<code>select ( int idx, bool single=true )</code>
<code>void</code>	<code>unselect ( int idx )</code>
<code>bool</code>	<code>is_selected ( int idx ) const</code>
<code>int</code>	<code>get_item_count ( ) const</code>

Continued on next page

Table 9.12 – continued from previous page

void	<code>remove_item( int idx )</code>
void	<code>clear()</code>
void	<code>sort_items_by_text()</code>
void	<code>set_fixed_column_width( int width )</code>
int	<code>get_fixed_column_width() const</code>
void	<code>set_max_text_lines( int lines )</code>
int	<code>get_max_text_lines() const</code>
void	<code>set_max_columns( int amount )</code>
int	<code>get_max_columns() const</code>
void	<code>set_select_mode( int mode )</code>
int	<code>get_select_mode() const</code>
void	<code>set_icon_mode( int mode )</code>
int	<code>get_icon_mode() const</code>
void	<code>set_min_icon_size( Vector2 size )</code>
Vector2	<code>get_min_icon_size() const</code>
void	<code>ensure_current_is_visible()</code>

### 9.132.3 Signals

- `item_activated( int index )`
- `multi_selected( int index, bool selected )`
- `item_selected( int index )`

### 9.132.4 Numeric Constants

- `ICON_MODE_TOP = 0`
- `ICON_MODE_LEFT = 1`
- `SELECT_SINGLE = 0`
- `SELECT_MULTI = 1`

### 9.132.5 Member Function Description

- void `add_item( String text, Texture icon=Object(), bool selectable=true )`
- void `add_icon_item( Texture icon, bool selectable=true )`
- void `set_item_text( int idx, String text )`
- `String get_item_text( int idx ) const`
- void `set_item_icon( int idx, Texture icon )`
- `Texture get_item_icon( int idx ) const`
- void `set_itemSelectable( int idx, bool selectable )`
- `bool is_itemSelectable( int idx ) const`
- void `set_itemDisabled( int idx, bool disabled )`
- `bool is_itemDisabled( int idx ) const`

- void **set\_item\_metadata** ( *int* idx, var metadata )
- void **get\_item\_metadata** ( *int* idx ) const
- void **set\_item\_custom\_bg\_color** ( *int* idx, *Color* custom\_bg\_color )
- *Color* **get\_item\_custom\_bg\_color** ( *int* idx ) const
- void **set\_item\_tooltip** ( *int* idx, *String* tooltip )
- *String* **get\_item\_tooltip** ( *int* idx ) const
- void **select** ( *int* idx, *bool* single=true )
- void **unselect** ( *int* idx )
- *bool* **is\_selected** ( *int* idx ) const
- *int* **get\_item\_count** ( ) const
- void **remove\_item** ( *int* idx )
- void **clear** ( )
- void **sort\_items\_by\_text** ( )
- void **set\_fixed\_column\_width** ( *int* width )
- *int* **get\_fixed\_column\_width** ( ) const
- void **set\_max\_text\_lines** ( *int* lines )
- *int* **get\_max\_text\_lines** ( ) const
- void **set\_max\_columns** ( *int* amount )
- *int* **get\_max\_columns** ( ) const
- void **set\_select\_mode** ( *int* mode )
- *int* **get\_select\_mode** ( ) const
- void **set\_icon\_mode** ( *int* mode )
- *int* **get\_icon\_mode** ( ) const
- void **set\_min\_icon\_size** ( *Vector2* size )
- *Vector2* **get\_min\_icon\_size** ( ) const
- void **ensure\_current\_is\_visible** ( )

## 9.133 Joint

Inherits: *Spatial* < *Node* < *Object*

Inherited By: *ConeTwistJoint*, *SliderJoint*, *Generic6DOFJoint*, *HingeJoint*, *PinJoint*

Category: Core

### 9.133.1 Brief Description

### 9.133.2 Member Functions

void	<code>set_node_a ( NodePath node )</code>
<i>NodePath</i>	<code>get_node_a () const</code>
void	<code>set_node_b ( NodePath node )</code>
<i>NodePath</i>	<code>get_node_b () const</code>
void	<code>set_solver_priority ( int priority )</code>
<i>int</i>	<code>get_solver_priority () const</code>
void	<code>set_exclude_nodes_from_collision ( bool enable )</code>
<i>bool</i>	<code>get_exclude_nodes_from_collision () const</code>

### 9.133.3 Member Function Description

- void `set_node_a ( NodePath node )`
- *NodePath* `get_node_a () const`
- void `set_node_b ( NodePath node )`
- *NodePath* `get_node_b () const`
- void `set_solver_priority ( int priority )`
- *int* `get_solver_priority () const`
- void `set_exclude_nodes_from_collision ( bool enable )`
- *bool* `get_exclude_nodes_from_collision () const`

## 9.134 Joint2D

**Inherits:** `Node2D < CanvasItem < Node < Object`

**Inherited By:** `PinJoint2D, DampedSpringJoint2D, GrooveJoint2D`

**Category:** Core

### 9.134.1 Brief Description

Base node for all joint constraints in 2D physics.

### 9.134.2 Member Functions

void	<code>set_node_a ( NodePath node )</code>
<i>NodePath</i>	<code>get_node_a () const</code>
void	<code>set_node_b ( NodePath node )</code>
<i>NodePath</i>	<code>get_node_b () const</code>
void	<code>set_bias ( float bias )</code>
<i>float</i>	<code>get_bias () const</code>
void	<code>set_exclude_nodes_from_collision ( bool enable )</code>
<i>bool</i>	<code>get_exclude_nodes_from_collision () const</code>

### 9.134.3 Description

Base node for all joint constraints in 2D physics. Joints take 2 bodies and apply a custom constraint.

### 9.134.4 Member Function Description

- void `set_node_a ( NodePath node )`

Set the path to the A node for the joint. Must be of type *PhysicsBody2D*.

- `NodePath get_node_a () const`

Return the path to the A node for the joint.

- void `set_node_b ( NodePath node )`

Set the path to the B node for the joint. Must be of type *PhysicsBody2D*.

- `NodePath get_node_b () const`

Return the path to the B node for the joint.

- void `set_bias ( float bias )`

- `float get_bias () const`

- void `set_exclude_nodes_from_collision ( bool enable )`

- `bool get_exclude_nodes_from_collision () const`

## 9.135 KinematicBody

**Inherits:** *PhysicsBody* < *CollisionObject* < *Spatial* < *Node* < *Object*

**Category:** Core

## 9.135.1 Brief Description

## 9.135.2 Member Functions

<code>Vector3</code>	<code>move ( Vector3 rel_vec )</code>
<code>Vector3</code>	<code>move_to ( Vector3 position )</code>
<code>bool</code>	<code>can_teleport_to ( Vector3 position )</code>
<code>bool</code>	<code>is_colliding () const</code>
<code>Vector3</code>	<code>get_collision_pos () const</code>
<code>Vector3</code>	<code>get_collision_normal () const</code>
<code>Vector3</code>	<code>get_collider_velocity () const</code>
<code>Object</code>	<code>get_collider () const</code>
<code>int</code>	<code>get_collider_shape () const</code>
<code>void</code>	<code>set_collide_with_static_bodies ( bool enable )</code>
<code>bool</code>	<code>can_collide_with_static_bodies () const</code>
<code>void</code>	<code>set_collide_with_kinematic_bodies ( bool enable )</code>
<code>bool</code>	<code>can_collide_with_kinematic_bodies () const</code>
<code>void</code>	<code>set_collide_with_rigid_bodies ( bool enable )</code>
<code>bool</code>	<code>can_collide_with_rigid_bodies () const</code>
<code>void</code>	<code>set_collide_with_character_bodies ( bool enable )</code>
<code>bool</code>	<code>can_collide_with_character_bodies () const</code>
<code>void</code>	<code>set_collision_margin ( float pixels )</code>
<code>float</code>	<code>get_collision_margin () const</code>

## 9.135.3 Member Function Description

- `Vector3 move ( Vector3 rel_vec )`
- `Vector3 move_to ( Vector3 position )`
- `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.

- `bool is_colliding () const`
- `Vector3 get_collision_pos () const`
- `Vector3 get_collision_normal () const`
- `Vector3 get_collider_velocity () const`
- `Object get_collider () const`
- `int get_collider_shape () const`
- `void set_collide_with_static_bodies ( bool enable )`
- `bool can_collide_with_static_bodies () const`
- `void set_collide_with_kinematic_bodies ( bool enable )`
- `bool can_collide_with_kinematic_bodies () const`
- `void set_collide_with_rigid_bodies ( bool enable )`
- `bool can_collide_with_rigid_bodies () const`
- `void set_collide_with_character_bodies ( bool enable )`

- `bool can_collide_with_character_bodies () const`
- `void set_collision_margin (float pixels)`
- `float get_collision_margin () const`

## 9.136 KinematicBody2D

Inherits: `PhysicsBody2D < CollisionObject2D < Node2D < CanvasItem < Node < Object`

Category: Core

### 9.136.1 Brief Description

Kinematic body 2D node.

### 9.136.2 Member Functions

<code>Vector2</code>	<code>move ( Vector2 rel_vec )</code>
<code>Vector2</code>	<code>move_to ( Vector2 position )</code>
<code>bool</code>	<code>test_move ( Vector2 rel_vec )</code>
<code>Vector2</code>	<code>get_travel () const</code>
<code>void</code>	<code>revert_motion ()</code>
<code>bool</code>	<code>is_colliding () const</code>
<code>Vector2</code>	<code>get_collision_pos () const</code>
<code>Vector2</code>	<code>get_collision_normal () const</code>
<code>Vector2</code>	<code>get.collider_velocity () const</code>
<code>Object</code>	<code>get.collider () const</code>
<code>int</code>	<code>get.collider_shape () const</code>
<code>Variant</code>	<code>get.collider_metadata () const</code>
<code>void</code>	<code>set_collision_margin (<code>float</code> pixels)</code>
<code>float</code>	<code>get_collision_margin () const</code>

### 9.136.3 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.

## 9.136.4 Member Function Description

- `Vector2 move ( Vector2 rel_vec )`

Move the body in the given direction, stopping if there is an obstacle.

- `Vector2 move_to ( Vector2 position )`

Move the body to the given position. This is not a teleport, and the body will stop if there is an obstacle.

- `bool test_move ( Vector2 rel_vec )`

Return true if there would be a collision if the body moved in the given direction.

- `Vector2 get_travel ( ) const`

Return the last movement done by the body.

- `void revert_motion ( )`

Undo the last movement done by the body.

- `bool is_colliding ( ) const`

Return whether the body is colliding with another.

- `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`.

- `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.collider_velocity ( ) const`

Return the velocity of the body that collided with this one.

- `Object 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`.

- `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`.

- `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.

- `float get_collision_margin ( ) const`

Return the collision margin for this object.

## 9.137 Label

Inherits: [Control](#) < [CanvasItem](#) < [Node](#) < [Object](#)

Category: Core

### 9.137.1 Brief Description

Control that displays formatted text.

### 9.137.2 Member Functions

void	<code>set_align ( int align )</code>
int	<code>get_align () const</code>
void	<code>set_valign ( int valign )</code>
int	<code>get_valign () const</code>
void	<code>set_text ( String text )</code>
<i>String</i>	<code>get_text () const</code>
void	<code>set_automap ( bool enable )</code>
<i>bool</i>	<code>has_automap () const</code>
void	<code>set_clip_text ( bool enable )</code>
<i>bool</i>	<code>is_clipping_text () const</code>
void	<code>set_uppercase ( bool enable )</code>
<i>bool</i>	<code>is_uppercase () const</code>
int	<code>get_line_height () const</code>
int	<code>get_line_count () const</code>
int	<code>get_total_character_count () const</code>
void	<code>set_visible_characters ( int amount )</code>
int	<code>get_visible_characters () const</code>
void	<code>set_percent_visible ( float percent_visible )</code>
<i>float</i>	<code>get_percent_visible () const</code>
void	<code>set_lines_skipped ( int lines_skipped )</code>
<i>int</i>	<code>get_lines_skipped () const</code>
void	<code>set_max_lines_visible ( int lines_visible )</code>
<i>int</i>	<code>get_max_lines_visible () const</code>

### 9.137.3 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.

## 9.137.4 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.

## 9.137.5 Member Function Description

- `void set_align ( int align )`

Sets the alignment mode to any of the ALIGN\_\* enumeration values.

- `int get_align () const`

Return the alignment mode (any of the ALIGN\_\* enumeration values).

- `void set_valign ( int valign )`

Sets the vertical alignment mode to any of the VALIGN\_\* enumeration values.

- `int get_valign () const`

Return the vertical alignment mode (any of the VALIGN\_\* enumeration values).

- `void set_text ( String text )`

Set the label text. Text can contain newlines.

- `String get_text () const`

Return the label text. Text can contain newlines.

- `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.

- `bool has_autowrap () const`

Return the state of the *autowrap* mode (see `set_autowrap`).

- `void set_clip_text ( bool enable )`

Cuts off the rest of the text if it is too wide.

- `bool is_clipping_text () const`

Return true if text would be cut off if it is too wide.

- `void set_uppercase ( bool enable )`

Display text in all capitals.

- `bool is_uppercase () const`

Return true if text is displayed in all capitals.

- `int get_line_height () const`

Return the height of a line.

- `int get_line_count () const`

Return the amount of lines.

- `int get_total_character_count () const`

Return the total length of the text.

- `void set_visible_characters ( int amount )`

Restricts the number of characters to display. Set to -1 to disable.

- `int get_visible_characters () const`

Return the restricted number of characters to display. Returns -1 if unrestricted.

- `void set_percent_visible ( float percent_visible )`

Restricts the number of characters to display (as a percentage of the total text).

- `float get_percent_visible () const`

Return the restricted number of characters to display (as a percentage of the total text).

- `void set_lines_skipped ( int lines_skipped )`

Sets the number of lines to skip before displaying. Useful for scrolling text.

- `int get_lines_skipped () const`

Return the the number of lines to skipped before displaying.

- `void set_max_lines_visible ( int lines_visible )`

Restricts the number of lines to display. Set to -1 to disable.

- `int get_max_lines_visible () const`

Return the restricted number of lines to display. Returns -1 if unrestricted.

## 9.138 LargeTexture

**Inherits:** `Texture < Resource < Reference < Object`

**Category:** Core

### 9.138.1 Brief Description

### 9.138.2 Member Functions

<code>int</code>	<code>add_piece ( Vector2 ofs, Texture texture )</code>
<code>void</code>	<code>set_piece_offset ( int idx, Vector2 ofs )</code>
<code>void</code>	<code>set_piece_texture ( int idx, Texture texture )</code>
<code>void</code>	<code>set_size ( Vector2 size )</code>
<code>void</code>	<code>clear ()</code>
<code>int</code>	<code>get_piece_count () const</code>
<code>Vector2</code>	<code>get_piece_offset ( int idx ) const</code>
<code>Texture</code>	<code>get_piece_texture ( int idx ) const</code>

### 9.138.3 Member Function Description

- `int add_piece ( Vector2 ofs, Texture texture )`
- `void set_piece_offset ( int idx, Vector2 ofs )`

- void `set_piece_texture` ( `int` idx, `Texture` texture )
- void `set_size` ( `Vector2` size )
- void `clear` ( )
- `int` `get_piece_count` ( ) const
- `Vector2` `get_piece_offset` ( `int` idx ) const
- `Texture` `get_piece_texture` ( `int` idx ) const

## 9.139 Light

**Inherits:** `VisualInstance < Spatial < Node < Object`

**Inherited By:** `SpotLight, OmniLight, DirectionalLight`

**Category:** Core

### 9.139.1 Brief Description

Provides a base class for different kinds of light nodes.

### 9.139.2 Member Functions

void	<code>set_parameter</code> ( <code>int</code> variable, <code>float</code> value )
<code>float</code>	<code>get_parameter</code> ( <code>int</code> variable ) const
void	<code>set_color</code> ( <code>int</code> color, <code>Color</code> value )
<code>Color</code>	<code>get_color</code> ( <code>int</code> color ) const
void	<code>set_project_shadows</code> ( <code>bool</code> enable )
<code>bool</code>	<code>has_project_shadows</code> ( ) const
void	<code>set_projector</code> ( <code>Texture</code> projector )
<code>Texture</code>	<code>get_projector</code> ( ) const
void	<code>set_operator</code> ( <code>int</code> operator )
<code>int</code>	<code>get_operator</code> ( ) const
void	<code>set_bake_mode</code> ( <code>int</code> bake_mode )
<code>int</code>	<code>get_bake_mode</code> ( ) const
void	<code>set_enabled</code> ( <code>bool</code> enabled )
<code>bool</code>	<code>is_enabled</code> ( ) const
void	<code>set_editor_only</code> ( <code>bool</code> editor_only )
<code>bool</code>	<code>is_editor_only</code> ( ) const

### 9.139.3 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**

## 9.139.4 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.

## 9.139.5 Member Function Description

- void **set\_parameter** ( *int* variable, *float* value )
- *float* **get\_parameter** ( *int* variable ) const
- void **set\_color** ( *int* color, *Color* value )
- *Color* **get\_color** ( *int* color ) const
- void **set\_project\_shadows** ( *bool* enable )
- *bool* **has\_project\_shadows** ( ) const
- void **set\_projector** ( *Texture* projector )
- *Texture* **get\_projector** ( ) const
- void **set\_operator** ( *int* operator )
- *int* **get\_operator** ( ) const
- void **set\_bake\_mode** ( *int* bake\_mode )
- *int* **get\_bake\_mode** ( ) const
- void **set\_enabled** ( *bool* enabled )
- *bool* **is\_enabled** ( ) const
- void **set\_editor\_only** ( *bool* editor\_only )
- *bool* **is\_editor\_only** ( ) const

## 9.140 Light2D

**Inherits:** *Node2D < CanvasItem < Node < Object*

**Category:** Core

## 9.140.1 Brief Description

## 9.140.2 Member Functions

void	<code>set_enabled ( bool enabled )</code>
<i>bool</i>	<code>is_enabled () const</code>
void	<code>set_texture ( Object texture )</code>
<i>Object</i>	<code>get_texture () const</code>
void	<code>set_texture_offset ( Vector2 texture_offset )</code>
<i>Vector2</i>	<code>get_texture_offset () const</code>
void	<code>set_color ( Color color )</code>
<i>Color</i>	<code>get_color () const</code>
void	<code>set_height ( float height )</code>
<i>float</i>	<code>get_height () const</code>
void	<code>set_energy ( float energy )</code>
<i>float</i>	<code>get_energy () const</code>
void	<code>set_texture_scale ( float texture_scale )</code>
<i>float</i>	<code>get_texture_scale () const</code>
void	<code>set_z_range_min ( int z )</code>
<i>int</i>	<code>get_z_range_min () const</code>
void	<code>set_z_range_max ( int z )</code>
<i>int</i>	<code>get_z_range_max () const</code>
void	<code>set_layer_range_min ( int layer )</code>
<i>int</i>	<code>get_layer_range_min () const</code>
void	<code>set_layer_range_max ( int layer )</code>
<i>int</i>	<code>get_layer_range_max () const</code>
void	<code>set_item_mask ( int item_mask )</code>
<i>int</i>	<code>get_item_mask () const</code>
void	<code>set_item_shadow_mask ( int item_shadow_mask )</code>
<i>int</i>	<code>get_item_shadow_mask () const</code>
void	<code>set_mode ( int mode )</code>
<i>int</i>	<code>get_mode () const</code>
void	<code>set_shadow_enabled ( bool enabled )</code>
<i>bool</i>	<code>is_shadow_enabled () const</code>
void	<code>set_shadow_buffer_size ( int size )</code>
<i>int</i>	<code>get_shadow_buffer_size () const</code>
void	<code>set_shadow_esm_multiplier ( float multiplier )</code>
<i>float</i>	<code>get_shadow_esm_multiplier () const</code>
void	<code>set_shadow_color ( Color shadow_color )</code>
<i>Color</i>	<code>get_shadow_color () const</code>

## 9.140.3 Numeric Constants

- **MODE\_ADD = 0**
- **MODE\_SUB = 1**
- **MODE\_MIX = 2**
- **MODE\_MASK = 3**

## 9.140.4 Member Function Description

- void **set\_enabled** ( *bool* enabled )
- *bool* **is\_enabled** ( ) const
- void **set\_texture** ( *Object* texture )
- *Object* **get\_texture** ( ) const
- void **set\_texture\_offset** ( *Vector2* texture\_offset )
- *Vector2* **get\_texture\_offset** ( ) const
- void **set\_color** ( *Color* color )
- *Color* **get\_color** ( ) const
- void **set\_height** ( *float* height )
- *float* **get\_height** ( ) const
- void **set\_energy** ( *float* energy )
- *float* **get\_energy** ( ) const
- void **set\_texture\_scale** ( *float* texture\_scale )
- *float* **get\_texture\_scale** ( ) const
- void **set\_z\_range\_min** ( *int* z )
- *int* **get\_z\_range\_min** ( ) const
- void **set\_z\_range\_max** ( *int* z )
- *int* **get\_z\_range\_max** ( ) const
- void **set\_layer\_range\_min** ( *int* layer )
- *int* **get\_layer\_range\_min** ( ) const
- void **set\_layer\_range\_max** ( *int* layer )
- *int* **get\_layer\_range\_max** ( ) const
- void **set\_item\_mask** ( *int* item\_mask )
- *int* **get\_item\_mask** ( ) const
- void **set\_item\_shadow\_mask** ( *int* item\_shadow\_mask )
- *int* **get\_item\_shadow\_mask** ( ) const
- void **set\_mode** ( *int* mode )
- *int* **get\_mode** ( ) const
- void **set\_shadow\_enabled** ( *bool* enabled )
- *bool* **is\_shadow\_enabled** ( ) const
- void **set\_shadow\_buffer\_size** ( *int* size )
- *int* **get\_shadow\_buffer\_size** ( ) const
- void **set\_shadow\_esm\_multiplier** ( *float* multiplier )
- *float* **get\_shadow\_esm\_multiplier** ( ) const
- void **set\_shadow\_color** ( *Color* shadow\_color )

- `Color get_shadow_color () const`

## 9.141 LightOccluder2D

Inherits: `Node2D < CanvasItem < Node < Object`

Category: Core

### 9.141.1 Brief Description

### 9.141.2 Member Functions

<code>void</code>	<code>set_occluder_polygon ( OccluderPolygon2D polygon )</code>
<code>OccluderPolygon2D</code>	<code>get_occluder_polygon () const</code>
<code>void</code>	<code>set_occluder_light_mask ( int mask )</code>
<code>int</code>	<code>get_occluder_light_mask () const</code>

### 9.141.3 Member Function Description

- `void set_occluder_polygon ( OccluderPolygon2D polygon )`
- `OccluderPolygon2D get_occluder_polygon () const`
- `void set_occluder_light_mask ( int mask )`
- `int get_occluder_light_mask () const`

## 9.142 LineEdit

Inherits: `Control < CanvasItem < Node < Object`

Category: Core

### 9.142.1 Brief Description

Control that provides single line string editing.

## 9.142.2 Member Functions

void	<code>set_align ( int align )</code>
<i>int</i>	<code>get_align () const</code>
void	<code>clear ()</code>
void	<code>select_all ()</code>
void	<code>set_text ( String text )</code>
<i>String</i>	<code>get_text () const</code>
void	<code>set_cursor_pos ( int pos )</code>
<i>int</i>	<code>get_cursor_pos () const</code>
void	<code>set_max_length ( int chars )</code>
<i>int</i>	<code>get_max_length () const</code>
void	<code>append_at_cursor ( String text )</code>
void	<code>set_editable ( bool enabled )</code>
<i>bool</i>	<code>is_editable () const</code>
void	<code>set_secret ( bool enabled )</code>
<i>bool</i>	<code>is_secret () const</code>
void	<code>select ( int from=0, int to=-1 )</code>

## 9.142.3 Signals

- `text_entered ( String text )`
- `text_changed ( String text )`

## 9.142.4 Numeric Constants

- `ALIGN_LEFT = 0`
- `ALIGN_CENTER = 1`
- `ALIGN_RIGHT = 2`
- `ALIGN_FILL = 3`

## 9.142.5 Description

LineEdit provides a single line string editor, used for text fields.

## 9.142.6 Member Function Description

- void `set_align ( int align )`
- *int* `get_align () const`
- void `clear ()`

Clear the *LineEdit* text.

- void `select_all ()`

Select the whole string.

- void `set_text ( String text )`

Set the text in the [LineEdit](#), clearing the existing one and the selection.

- `String get_text () const`

Return the text in the [LineEdit](#).

- `void set_cursor_pos ( int pos )`

Set the cursor position inside the [LineEdit](#), causing it to scroll if needed.

- `int get_cursor_pos () const`

Return the cursor position inside the [LineEdit](#).

- `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.

- `int get_max_length () const`

Return the maximum amount of characters the [LineEdit](#) can edit. If 0 is returned, no limit exists.

- `void append_at_cursor ( String text )`

Append text at cursor, scrolling the [LineEdit](#) when 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.

- `bool is_editable () const`

Return the *editable* status of the [LineEdit](#) (see `set_editable`).

- `void set_secret ( bool enabled )`

Set the *secret* status of the [LineEdit](#). When enabled, every character is displayed as “\*”.

- `bool is_secret () const`

Return the *secret* status of the [LineEdit](#) (see `set_secret`).

- `void select ( int from=0, int to=-1 )`

## 9.143 LineShape2D

**Inherits:** [Shape2D](#) < [Resource](#) < [Reference](#) < [Object](#)

**Category:** Core

### 9.143.1 Brief Description

Line shape for 2D collision objects.

### 9.143.2 Member Functions

<code>void</code>	<code>set_normal ( Vector2 normal )</code>
<code>Vector2</code>	<code>get_normal () const</code>
<code>void</code>	<code>set_d ( float d )</code>
<code>float</code>	<code>get_d () const</code>

### 9.143.3 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.

### 9.143.4 Member Function Description

- void `set_normal ( Vector2 normal )`

Set the line normal.

- `Vector2 get_normal () const`

Return the line normal.

- void `set_d ( float d )`

Set the line distance from the origin.

- `float get_d () const`

Return the line distance from the origin.

## 9.144 MainLoop

**Inherits:** `Object`

**Inherited By:** `SceneTree`

**Category:** Core

### 9.144.1 Brief Description

Main loop is the abstract main loop base class.

### 9.144.2 Member Functions

void	<code>_finalize () virtual</code>
void	<code>_idle ( float delta ) virtual</code>
void	<code>_initialize () virtual</code>
void	<code>_input_event ( InputEvent ev ) virtual</code>
void	<code>_input_text ( String text ) virtual</code>
void	<code>_iteration ( float delta ) virtual</code>
void	<code>input_event ( InputEvent ev )</code>
void	<code>input_text ( String text )</code>
void	<code>init ()</code>
bool	<code>iteration ( float delta )</code>
bool	<code>idle ( float delta )</code>
void	<code>finish ()</code>

### 9.144.3 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

### 9.144.4 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 SceneMainLoop is created), unless a main [Script](#) is supplied, which may or not create and return a [MainLoop](#).

### 9.144.5 Member Function Description

- 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 **input\_event** (*InputEvent* ev)
- void **input\_text** (*String* text)
- void **init** ()
- *bool* **iteration** (*float* delta)
- *bool* **idle** (*float* delta)
- void **finish** ()

## 9.145 MarginContainer

**Inherits:** [Container](#) < [Control](#) < [CanvasItem](#) < [Node](#) < [Object](#)

**Category:** Core

### 9.145.1 Brief Description

Simple margin container.

## 9.145.2 Description

Simple margin container. Adds a left margin to anything contained.

# 9.146 Marshalls

**Inherits:** [Reference](#) < [Object](#)

**Category:** Core

## 9.146.1 Brief Description

## 9.146.2 Member Functions

<i>String</i>	<code>variant_to_base64 ( var variant )</code>
<i>Variant</i>	<code>base64_to_variant ( <i>String</i> base64_str )</code>
<i>String</i>	<code>raw_to_base64 ( <i>RawArray</i> array )</code>
<i>RawArray</i>	<code>base64_to_raw ( <i>String</i> base64_str )</code>
<i>String</i>	<code>utf8_to_base64 ( <i>String</i> utf8_str )</code>
<i>String</i>	<code>base64_to_utf8 ( <i>String</i> base64_str )</code>

## 9.146.3 Member Function Description

- *String* **variant\_to\_base64** ( var variant )
- Variant **base64\_to\_variant** ( *String* base64\_str )
- *String* **raw\_to\_base64** ( *RawArray* array )
- *RawArray* **base64\_to\_raw** ( *String* base64\_str )
- *String* **utf8\_to\_base64** ( *String* utf8\_str )
- *String* **base64\_to\_utf8** ( *String* base64\_str )

# 9.147 Material

**Inherits:** [Resource](#) < [Reference](#) < [Object](#)

**Inherited By:** [ShaderMaterial](#), [FixedMaterial](#)

**Category:** Core

## 9.147.1 Brief Description

Abstract base *Resource* for coloring and shading geometry.

## 9.147.2 Member Functions

<code>void</code>	<code>set_flag ( int flag, bool enable )</code>
<code>bool</code>	<code>get_flag ( int flag ) const</code>
<code>void</code>	<code>set_blend_mode ( int mode )</code>
<code>int</code>	<code>get_blend_mode ( ) const</code>
<code>void</code>	<code>set_line_width ( float width )</code>
<code>float</code>	<code>get_line_width ( ) const</code>
<code>void</code>	<code>set_depth_draw_mode ( int mode )</code>
<code>int</code>	<code>get_depth_draw_mode ( ) const</code>

## 9.147.3 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**

## 9.147.4 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.

## 9.147.5 Member Function Description

- 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.

- `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.

- 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.

- `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.

- 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).

- `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_depth_draw_mode` ( `int` mode )

- `int get_depth_draw_mode` ( ) const

## 9.148 MaterialShader

**Inherits:** [Shader](#) < [Resource](#) < [Reference](#) < [Object](#)

**Category:** Core

### 9.148.1 Brief Description

## 9.149 MaterialShaderGraph

**Inherits:** [ShaderGraph](#) < [Shader](#) < [Resource](#) < [Reference](#) < [Object](#)

**Category:** Core

### 9.149.1 Brief Description

## 9.150 Matrix3

**Category:** Built-In Types

### 9.150.1 Brief Description

3x3 matrix datatype.

### 9.150.2 Member Functions

<i>float</i>	<i>determinant ()</i>
<i>Vector3</i>	<i>get_euler ()</i>
<i>int</i>	<i>get_orthogonal_index ()</i>
<i>Vector3</i>	<i>get_scale ()</i>
<i>Matrix3</i>	<i>inverse ()</i>
<i>Matrix3</i>	<i>orthonormalized ()</i>
<i>Matrix3</i>	<i>rotated ( Vector3 axis, float phi )</i>
<i>Matrix3</i>	<i>scaled ( Vector3 scale )</i>
<i>float</i>	<i>tdotx ( Vector3 with )</i>
<i>float</i>	<i>tdoty ( Vector3 with )</i>
<i>float</i>	<i>tdotz ( Vector3 with )</i>
<i>Matrix3</i>	<i>transposed ()</i>
<i>Vector3</i>	<i>xform ( Vector3 v )</i>
<i>Vector3</i>	<i>xform_inv ( Vector3 v )</i>
<i>Matrix3</i>	<i>Matrix3 ( Vector3 x_axis, Vector3 y_axis, Vector3 z_axis )</i>
<i>Matrix3</i>	<i>Matrix3 ( Vector3 axis, float phi )</i>
<i>Matrix3</i>	<i>Matrix3 ( Quat from )</i>

### 9.150.3 Member Variables

- *Vector3 x*
- *Vector3 y*
- *Vector3 z*

### 9.150.4 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*.

### 9.150.5 Member Function Description

- *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.

- *Vector3 xform ( Vector3 v )*

Return a vector transformed by the matrix and return it.

- *Vector3 xform\_inv ( Vector3 v )*

Return a vector transformed by the transposed matrix and return it.

- *Matrix3 Matrix3 ( Vector3 x\_axis, Vector3 y\_axis, Vector3 z\_axis )*

Create a matrix from 3 axis vectors.

- *Matrix3 Matrix3 ( Vector3 axis, float phi )*

Create a matrix from an axis vector and an angle.

- *Matrix3 Matrix3 ( Quat from )*

Create a matrix from a quaternion.

## 9.151 Matrix32

**Category:** Built-In Types

### 9.151.1 Brief Description

3x2 Matrix for 2D transforms.

## 9.151.2 Member Functions

<i>Matrix32</i>	<code>affine_inverse ()</code>
<i>Matrix32</i>	<code>basis_xform ( var v )</code>
<i>Matrix32</i>	<code>basis_xform_inv ( var v )</code>
<i>Vector2</i>	<code>get_origin ()</code>
<i>float</i>	<code>get_rotation ()</code>
<i>Vector2</i>	<code>get_scale ()</code>
<i>Matrix32</i>	<code>interpolate_with ( Matrix32 m, float c )</code>
<i>Matrix32</i>	<code>inverse ()</code>
<i>Matrix32</i>	<code>orthonormalized ()</code>
<i>Matrix32</i>	<code>rotated ( float phi )</code>
<i>Matrix32</i>	<code>scaled ( Vector2 scale )</code>
<i>Matrix32</i>	<code>translated ( Vector2 offset )</code>
<i>Matrix32</i>	<code>xform ( var v )</code>
<i>Matrix32</i>	<code>xform_inv ( var v )</code>
<i>Matrix32</i>	<code>Matrix32 ( float rot, Vector2 pos )</code>
<i>Matrix32</i>	<code>Matrix32 ( Vector2 x_axis, Vector2 y_axis, Vector2 origin )</code>
<i>Matrix32</i>	<code>Matrix32 ( Transform from )</code>

## 9.151.3 Member Variables

- *Vector2 x*
- *Vector2 y*
- *Vector2 o*

## 9.151.4 Description

3x2 Matrix for 2D transforms.

## 9.151.5 Member Function Description

- *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 )
- *Matrix32* **Matrix32** ( *float* rot, *Vector2* pos )
- *Matrix32* **Matrix32** ( *Vector2* x\_axis, *Vector2* y\_axis, *Vector2* origin )
- *Matrix32* **Matrix32** ( *Transform* from )

## 9.152 MenuButton

**Inherits:** *Button* < *BaseButton* < *Control* < *CanvasItem* < *Node* < *Object*

**Category:** Core

### 9.152.1 Brief Description

Special button that brings up a *PopupMenu* when clicked.

### 9.152.2 Member Functions

<i>PopupMenu</i>	<i>get_popup()</i>
------------------	--------------------

### 9.152.3 Signals

- **about\_to\_show()**

### 9.152.4 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.

### 9.152.5 Member Function Description

- *PopupMenu* **get\_popup()**

Return the *PopupMenu* contained in this button.

## 9.153 Mesh

**Inherits:** *Resource* < *Reference* < *Object*

**Category:** Core

### 9.153.1 Brief Description

A *Resource* that contains vertex-array based geometry.

## 9.153.2 Member Functions

void	<code>add_morph_target ( String name )</code>
<i>int</i>	<code>get_morph_target_count () const</code>
<i>String</i>	<code>get_morph_target_name ( int index ) const</code>
void	<code>clear_morph_targets ()</code>
void	<code>set_morph_target_mode ( int mode )</code>
<i>int</i>	<code>get_morph_target_mode () const</code>
void	<code>add_surface ( int primitive, Array arrays, Array morph_arrays=Array(), bool alphasort=false )</code>
<i>int</i>	<code>get_surface_count () const</code>
void	<code>surface_remove ( int surf_idx )</code>
<i>int</i>	<code>surface_get_array_len ( int surf_idx ) const</code>
<i>int</i>	<code>surface_get_array_index_len ( int surf_idx ) const</code>
<i>int</i>	<code>surface_get_format ( int surf_idx ) const</code>
<i>int</i>	<code>surface_get_primitive_type ( int surf_idx ) const</code>
void	<code>surface_set_material ( int surf_idx, Material material )</code>
<i>Material</i>	<code>surface_get_material ( int surf_idx ) const</code>
void	<code>surface_set_name ( int surf_idx, String name )</code>
<i>String</i>	<code>surface_get_name ( int surf_idx ) const</code>
void	<code>center_geometry ()</code>
void	<code>regen_normalmaps ()</code>
void	<code>set_custom_aabb ( AABB aabb )</code>
<i>AABB</i>	<code>get_custom_aabb () const</code>

## 9.153.3 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.

#### 9.153.4 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.

#### 9.153.5 Member Function Description

- void **add\_morph\_target** ( *String* name )
- *int* **get\_morph\_target\_count** ( ) const
- *String* **get\_morph\_target\_name** ( *int* index ) const
- void **clear\_morph\_targets** ( )
- void **set\_morph\_target\_mode** ( *int* mode )
- *int* **get\_morph\_target\_mode** ( ) const
- 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.

- *int* **get\_surface\_count** ( ) const

Return the amount of surfaces that the *Mesh* holds.

- void **surface\_remove** ( *int* surf\_idx )

Remove a surface at position surf\_idx, shifting greater surfaces one surf\_idx slot down.

- `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_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_format ( int surf_idx ) const`

Return the format mask of the requested surface (see `add_surface`).

- `int surface_get_primitive_type ( int surf_idx ) const`

Return the primitive type of the requested surface (see `add_surface`).

- `void surface_set_material ( int surf_idx, Material material )`

Set a `Material` for a given surface. Surface will be rendered using this material.

- `Material surface_get_material ( int surf_idx ) const`

Return a `Material` in a given surface. Surface is rendered using this material.

- `void surface_set_name ( int surf_idx, String name )`

- `String surface_get_name ( int surf_idx ) const`

- `void center_geometry ()`

- `void regen_normalmaps ()`

- `void set_custom_aabb ( AABB aabb )`

- `AABB get_custom_aabb () const`

## 9.154 MeshDataTool

**Inherits:** `Reference < Object`

**Category:** Core

### 9.154.1 Brief Description

### 9.154.2 Member Functions

<code>void</code>	<code>clear ()</code>
<code>int</code>	<code>create_from_surface ( Object mesh, int surface )</code>
<code>int</code>	<code>commit_to_surface ( Object mesh )</code>
<code>int</code>	<code>get_format () const</code>
<code>int</code>	<code>get_vertex_count () const</code>
<code>int</code>	<code>get_edge_count () const</code>
<code>int</code>	<code>get_face_count () const</code>
<code>void</code>	<code>set_vertex ( int idx, Vector3 vertex )</code>
<code>Vector3</code>	<code>get_vertex ( int idx ) const</code>
<code>void</code>	<code>set_vertex_normal ( int idx, Vector3 normal )</code>
<code>Vector3</code>	<code>get_vertex_normal ( int idx ) const</code>
<code>void</code>	<code>set_vertex_tangent ( int idx, Plane tangent )</code>

Continued on next page

Table 9.14 – continued from previous page

<i>Plane</i>	<code>get_vertex_tangent ( int idx ) const</code>
<code>void</code>	<code>set_vertex_uv ( int idx, Vector2 uv )</code>
<i>Vector2</i>	<code>get_vertex_uv ( int idx ) const</code>
<code>void</code>	<code>set_vertex_uv2 ( int idx, Vector2 uv2 )</code>
<i>Vector2</i>	<code>get_vertex_uv2 ( int idx ) const</code>
<code>void</code>	<code>set_vertex_color ( int idx, Color color )</code>
<i>Color</i>	<code>get_vertex_color ( int idx ) const</code>
<code>void</code>	<code>set_vertex_bones ( int idx, IntArray bones )</code>
<i>IntArray</i>	<code>get_vertex_bones ( int idx ) const</code>
<code>void</code>	<code>set_vertex_weights ( int idx, RealArray weights )</code>
<i>RealArray</i>	<code>get_vertex_weights ( int idx ) const</code>
<code>void</code>	<code>set_vertex_meta ( int idx, var meta )</code>
<code>void</code>	<code>get_vertex_meta ( int idx ) const</code>
<i>IntArray</i>	<code>get_vertex_edges ( int idx ) const</code>
<i>IntArray</i>	<code>get_vertex_faces ( int idx ) const</code>
<i>int</i>	<code>get_edge_vertex ( int idx, int vertex ) const</code>
<i>IntArray</i>	<code>get_edge_faces ( int idx ) const</code>
<code>void</code>	<code>set_edge_meta ( int idx, var meta )</code>
<code>void</code>	<code>get_edge_meta ( int idx ) const</code>
<i>int</i>	<code>get_face_vertex ( int idx, int vertex ) const</code>
<i>int</i>	<code>get_face_edge ( int idx, int edge ) const</code>
<code>void</code>	<code>set_face_meta ( int idx, var meta )</code>
<code>void</code>	<code>get_face_meta ( int idx ) const</code>
<i>Vector3</i>	<code>get_face_normal ( int idx ) const</code>
<code>void</code>	<code>set_material ( Material material )</code>
<i>Object</i>	<code>get_material () const</code>

### 9.154.3 Member Function Description

- `void clear ()`
- `int create_from_surface ( Object mesh, int surface )`
- `int commit_to_surface ( Object mesh )`
- `int get_format () const`
- `int get_vertex_count () const`
- `int get_edge_count () const`
- `int get_face_count () const`
- `void set_vertex ( int idx, Vector3 vertex )`
- `Vector3 get_vertex ( int idx ) const`
- `void set_vertex_normal ( int idx, Vector3 normal )`
- `Vector3 get_vertex_normal ( int idx ) const`
- `void set_vertex_tangent ( int idx, Plane tangent )`
- `Plane get_vertex_tangent ( int idx ) const`
- `void set_vertex_uv ( int idx, Vector2 uv )`
- `Vector2 get_vertex_uv ( int idx ) const`

- void **set\_vertex\_uv2** ( *int* idx, *Vector2* uv2 )
- *Vector2* **get\_vertex\_uv2** ( *int* idx ) const
- void **set\_vertex\_color** ( *int* idx, *Color* color )
- *Color* **get\_vertex\_color** ( *int* idx ) const
- void **set\_vertex\_bones** ( *int* idx, *IntArray* bones )
- *IntArray* **get\_vertex\_bones** ( *int* idx ) const
- void **set\_vertex\_weights** ( *int* idx, *RealArray* weights )
- *RealArray* **get\_vertex\_weights** ( *int* idx ) const
- void **set\_vertex\_meta** ( *int* idx, var meta )
- void **get\_vertex\_meta** ( *int* idx ) const
- *IntArray* **get\_vertex\_edges** ( *int* idx ) const
- *IntArray* **get\_vertex\_faces** ( *int* idx ) const
- *int* **get\_edge\_vertex** ( *int* idx, *int* vertex ) const
- *IntArray* **get\_edge\_faces** ( *int* idx ) const
- void **set\_edge\_meta** ( *int* idx, var meta )
- void **get\_edge\_meta** ( *int* idx ) const
- *int* **get\_face\_vertex** ( *int* idx, *int* vertex ) const
- *int* **get\_face\_edge** ( *int* idx, *int* edge ) const
- void **set\_face\_meta** ( *int* idx, var meta )
- void **get\_face\_meta** ( *int* idx ) const
- *Vector3* **get\_face\_normal** ( *int* idx ) const
- void **set\_material** ( *Material* material )
- *Object* **get\_material** ( ) const

## 9.155 MeshInstance

**Inherits:** *GeometryInstance* < *VisualInstance* < *Spatial* < *Node* < *Object*

**Category:** Core

### 9.155.1 Brief Description

Node that instances meshes into a Scenario.

## 9.155.2 Member Functions

void	<code>set_mesh ( Mesh mesh )</code>
<i>Mesh</i>	<code>get_mesh () const</code>
void	<code>set_skeleton_path ( NodePath skeleton_path )</code>
<i>NodePath</i>	<code>get_skeleton_path ()</code>
<i>AABB</i>	<code>get_aabb () const</code>
void	<code>create_trimesh_collision ()</code>
void	<code>create_convex_collision ()</code>

## 9.155.3 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.

## 9.155.4 Member Function Description

- void `set_mesh ( Mesh mesh )`

Set the *Mesh* resource for the instance.

- *Mesh* `get_mesh () const`

Return the current *Mesh* resource for the instance.

- void `set_skeleton_path ( NodePath skeleton_path )`
- *NodePath* `get_skeleton_path ()`
- *AABB* `get_aabb () const`

Return the AABB of the mesh, in local coordinates.

- void `create_trimesh_collision ()`

This helper creates a *StaticBody* child *Node* using the mesh geometry as collision. It's mainly used for testing.

- void `create_convex_collision ()`

## 9.156 MeshLibrary

**Inherits:** *Resource < Reference < Object*

**Category:** Core

### 9.156.1 Brief Description

Library of meshes.

## 9.156.2 Member Functions

void	<code>create_item ( int id )</code>
void	<code>set_item_name ( int id, String name )</code>
void	<code>set_item_mesh ( int id, Mesh mesh )</code>
void	<code>set_item_shape ( int id, Shape shape )</code>
<i>String</i>	<code>get_item_name ( int id ) const</code>
<i>Mesh</i>	<code>get_item_mesh ( int id ) const</code>
<i>Shape</i>	<code>get_item_shape ( int id ) const</code>
void	<code>remove_item ( int id )</code>
void	<code>clear ()</code>
<i>IntArray</i>	<code>get_item_list () const</code>
<i>int</i>	<code>get_last_unused_item_id () const</code>

## 9.156.3 Description

Library of meshes. Contains a list of *Mesh* resources, each with name and ID. Useful for GridMap or painting Terrain.

## 9.156.4 Member Function Description

- void **create\_item** ( *int* id )

Create a new item in the library, supplied an id.

- void **set\_item\_name** ( *int* id, *String* name )

Set the name of the item.

- void **set\_item\_mesh** ( *int* id, *Mesh* mesh )

Set the mesh of the item.

- void **set\_item\_shape** ( *int* id, *Shape* shape )

- *String* **get\_item\_name** ( *int* id ) const

Return the name of the item.

- *Mesh* **get\_item\_mesh** ( *int* id ) const

Return the mesh of the item.

- *Shape* **get\_item\_shape** ( *int* id ) const

- void **remove\_item** ( *int* id )

Remove the item.

- void **clear** ()

Clear the library.

- *IntArray* **get\_item\_list** () const

Return the list of items.

- *int* **get\_last\_unused\_item\_id** () const

Get an unused id for a new item.

## 9.157 MultiMesh

Inherits: [Resource](#) < [Reference](#) < [Object](#)

Category: Core

### 9.157.1 Brief Description

Provides high performance mesh instancing.

### 9.157.2 Member Functions

void	<code>set_mesh ( Mesh mesh )</code>
<i>Mesh</i>	<code>get_mesh () const</code>
void	<code>set_instance_count ( int count )</code>
<i>int</i>	<code>get_instance_count () const</code>
void	<code>set_instance_transform ( int instance, Transform transform )</code>
<i>Transform</i>	<code>get_instance_transform ( int instance ) const</code>
void	<code>set_instance_color ( int instance, Color color )</code>
<i>Color</i>	<code>get_instance_color ( int instance ) const</code>
void	<code>set_aabb ( AABB visibility_aabb )</code>
<i>AABB</i>	<code>get_aabb () const</code>
void	<code>generate_aabb ()</code>

### 9.157.3 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`.

### 9.157.4 Member Function Description

- void `set_mesh ( Mesh mesh )`

Set the [Mesh](#) resource to be drawn in multiple instances.

- [Mesh](#) `get_mesh () const`

Return the [Mesh](#) resource drawn as multiple instances.

- 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.

- [int](#) `get_instance_count () const`

Return the amount of instances that is going to be drawn.

- `void set_instance_transform ( int instance, Transform transform )`

Set the transform for a specific instance.

- `Transform get_instance_transform ( int instance ) const`

Return the transform of a specific instance.

- `void set_instance_color ( int instance, Color color )`

Set the color of a specific instance.

- `Color get_instance_color ( int instance ) const`

Get the color of a specific instance.

- `void set_aabb ( AABB visibility_aabb )`

Set the visibility AABB. If not provided, MultiMesh will not be visible.

- `AABB get_aabb () const`

Return the visibility AABB.

- `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.

## 9.158 MultiMeshInstance

**Inherits:** *GeometryInstance < VisualInstance < Spatial < Node < Object*

**Category:** Core

### 9.158.1 Brief Description

Node that instances a *MultiMesh*.

### 9.158.2 Member Functions

<code>void</code>	<code>set_multimesh ( Object multimesh )</code>
<code>Object</code>	<code>get_multimesh () const</code>

### 9.158.3 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).

## 9.158.4 Member Function Description

- void **set\_multimesh** ( *Object* multimesh )

Set the *MultiMesh* to be instance.

- *Object* **get\_multimesh** ( ) const

Return the *MultiMesh* that is used for instancing.

## 9.159 Mutex

**Inherits:** *Reference* < *Object*

**Category:** Core

### 9.159.1 Brief Description

### 9.159.2 Member Functions

void	<i>lock</i> ( )
Error	<i>try_lock</i> ( )
void	<i>unlock</i> ( )

### 9.159.3 Member Function Description

- void **lock** ( )
- Error **try\_lock** ( )
- void **unlock** ( )

## 9.160 Navigation

**Inherits:** *Spatial* < *Node* < *Object*

**Category:** Core

## 9.160.1 Brief Description

## 9.160.2 Member Functions

<i>int</i>	<code>navmesh_create ( NavigationMesh mesh, Transform xform, Object owner=NULL )</code>
<code>void</code>	<code>navmesh_set_transform ( int id, Transform xform )</code>
<code>void</code>	<code>navmesh_remove ( int id )</code>
<code>Vector3Array</code>	<code>get_simple_path ( Vector3 start, Vector3 end, bool optimize=true )</code>
<code>Vector3</code>	<code>get_closest_point_to_segment ( Vector3 start, Vector3 end, bool use_collision=false )</code>
<code>Vector3</code>	<code>get_closest_point ( Vector3 to_point )</code>
<code>Vector3</code>	<code>get_closest_point_normal ( Vector3 to_point )</code>
<code>Object</code>	<code>get_closest_point_owner ( Vector3 to_point )</code>
<code>void</code>	<code>set_up_vector ( Vector3 up )</code>
<code>Vector3</code>	<code>get_up_vector ( ) const</code>

## 9.160.3 Member Function Description

- `int navmesh_create ( NavigationMesh mesh, Transform xform, Object owner=NULL )`
- `void navmesh_set_transform ( int id, Transform xform )`
- `void navmesh_remove ( int id )`
- `Vector3Array get_simple_path ( Vector3 start, Vector3 end, bool optimize=true )`
- `Vector3 get_closest_point_to_segment ( Vector3 start, Vector3 end, bool use_collision=false )`
- `Vector3 get_closest_point ( Vector3 to_point )`
- `Vector3 get_closest_point_normal ( Vector3 to_point )`
- `Object get_closest_point_owner ( Vector3 to_point )`
- `void set_up_vector ( Vector3 up )`
- `Vector3 get_up_vector ( ) const`

## 9.161 Navigation2D

Inherits: `Node2D < CanvasItem < Node < Object`

Category: Core

## 9.161.1 Brief Description

## 9.161.2 Member Functions

<i>int</i>	<code>navpoly_create ( NavigationPolygon mesh, Matrix32 xform, Object owner=NULL )</code>
<code>void</code>	<code>navpoly_set_transform ( int id, Matrix32 xform )</code>
<code>void</code>	<code>navpoly_remove ( int id )</code>
<code>Vector2Array</code>	<code>get_simple_path ( Vector2 start, Vector2 end, bool optimize=true )</code>
<code>Vector2</code>	<code>get_closest_point ( Vector2 to_point )</code>
<code>Object</code>	<code>get_closest_point_owner ( Vector2 to_point )</code>

### 9.161.3 Member Function Description

- `int navpoly_create ( NavigationPolygon mesh, Matrix32 xform, Object owner=NULL )`
- `void navpoly_set_transform ( int id, Matrix32 xform )`
- `void navpoly_remove ( int id )`
- `Vector2Array get_simple_path ( Vector2 start, Vector2 end, bool optimize=true )`
- `Vector2 get_closest_point ( Vector2 to_point )`
- `Object get_closest_point_owner ( Vector2 to_point )`

## 9.162 NavigationMesh

**Inherits:** `Resource < Reference < Object`

**Category:** Core

### 9.162.1 Brief Description

### 9.162.2 Member Functions

void	<code>set_vertices ( Vector3Array vertices )</code>
<code>Vector3Array</code>	<code>get_vertices () const</code>
void	<code>add_polygon ( IntArray polygon )</code>
<code>int</code>	<code>get_polygon_count () const</code>
<code>IntArray</code>	<code>get_polygon ( int idx )</code>
void	<code>clear_polygons ()</code>

### 9.162.3 Member Function Description

- `void set_vertices ( Vector3Array vertices )`
- `Vector3Array get_vertices () const`
- `void add_polygon ( IntArray polygon )`
- `int get_polygon_count () const`
- `IntArray get_polygon ( int idx )`
- `void clear_polygons ()`

## 9.163 NavigationMeshInstance

**Inherits:** `Spatial < Node < Object`

**Category:** Core

### 9.163.1 Brief Description

### 9.163.2 Member Functions

void	<code>set_navigation_mesh ( Object navmesh )</code>
<i>Object</i>	<code>get_navigation_mesh ( ) const</code>
void	<code>set_enabled ( bool enabled )</code>
<i>bool</i>	<code>is_enabled ( ) const</code>

### 9.163.3 Member Function Description

- void `set_navigation_mesh ( Object navmesh )`
- *Object* `get_navigation_mesh ( ) const`
- void `set_enabled ( bool enabled )`
- *bool* `is_enabled ( ) const`

## 9.164 NavigationPolygon

Inherits: *Resource* < *Reference* < *Object*

Category: Core

### 9.164.1 Brief Description

### 9.164.2 Member Functions

void	<code>set_vertices ( Vector2Array vertices )</code>
<i>Vector2Array</i>	<code>get_vertices ( ) const</code>
void	<code>add_polygon ( IntArray polygon )</code>
<i>int</i>	<code>get_polygon_count ( ) const</code>
<i>IntArray</i>	<code>get_polygon ( int idx )</code>
void	<code>clear_polygons ( )</code>
void	<code>add_outline ( Vector2Array outline )</code>
void	<code>add_outline_at_index ( Vector2Array outline, int index )</code>
<i>int</i>	<code>get_outline_count ( ) const</code>
void	<code>set_outline ( int idx, Vector2Array outline )</code>
<i>Vector2Array</i>	<code>get_outline ( int idx ) const</code>
void	<code>remove_outline ( int idx )</code>
void	<code>clear_outlines ( )</code>
void	<code>make_polygons_from_outlines ( )</code>

### 9.164.3 Member Function Description

- void `set_vertices ( Vector2Array vertices )`
- *Vector2Array* `get_vertices ( ) const`
- void `add_polygon ( IntArray polygon )`

- `int get_polygon_count () const`
- `IntArray get_polygon ( int idx )`
- `void clear_polygons ()`
- `void add_outline ( Vector2Array outline )`
- `void add_outline_at_index ( Vector2Array outline, int index )`
- `int get_outline_count () const`
- `void set_outline ( int idx, Vector2Array outline )`
- `Vector2Array get_outline ( int idx ) const`
- `void remove_outline ( int idx )`
- `void clear_outlines ()`
- `void make_polygons_from_outlines ()`

## 9.165 NavigationPolygonInstance

Inherits: `Node2D < CanvasItem < Node < Object`

Category: Core

### 9.165.1 Brief Description

### 9.165.2 Member Functions

<code>void</code>	<code>set_navigation_polygon ( Object navpoly )</code>
<code>Object</code>	<code>get_navigation_polygon () const</code>
<code>void</code>	<code>set_enabled ( bool enabled )</code>
<code>bool</code>	<code>is_enabled () const</code>

### 9.165.3 Member Function Description

- `void set_navigation_polygon ( Object navpoly )`
- `Object get_navigation_polygon () const`
- `void set_enabled ( bool enabled )`
- `bool is_enabled () const`

## 9.166 Nil

Category: Built-In Types

## 9.166.1 Brief Description

## 9.166.2 Member Functions

void	Nil ( <i>bool</i> from )
void	Nil ( <i>int</i> from )
void	Nil ( <i>float</i> from )
void	Nil ( <i>String</i> from )
void	Nil ( <i>Vector2</i> from )
void	Nil ( <i>Rect2</i> from )
void	Nil ( <i>Vector3</i> from )
void	Nil ( <i>Matrix32</i> from )
void	Nil ( <i>Plane</i> from )
void	Nil ( <i>Quat</i> from )
void	Nil ( <i>AABB</i> from )
void	Nil ( <i>Matrix3</i> from )
void	Nil ( <i>Transform</i> from )
void	Nil ( <i>Color</i> from )
void	Nil ( <i>Image</i> from )
void	Nil ( <i>NodePath</i> from )
void	Nil ( <i>RID</i> from )
void	Nil ( <i>Object</i> from )
void	Nil ( <i>InputEvent</i> from )
void	Nil ( <i>Dictionary</i> from )
void	Nil ( <i>Array</i> from )
void	Nil ( <i>RawArray</i> from )
void	Nil ( <i>IntArray</i> from )
void	Nil ( <i>RealArray</i> from )
void	Nil ( <i>StringArray</i> from )
void	Nil ( <i>Vector2Array</i> from )
void	Nil ( <i>Vector3Array</i> from )
void	Nil ( <i>ColorArray</i> from )

## 9.166.3 Member Function Description

- void **Nil** ( *bool* from )
- void **Nil** ( *int* from )
- void **Nil** ( *float* from )
- void **Nil** ( *String* from )
- void **Nil** ( *Vector2* from )
- void **Nil** ( *Rect2* from )
- void **Nil** ( *Vector3* from )
- void **Nil** ( *Matrix32* from )
- void **Nil** ( *Plane* from )
- void **Nil** ( *Quat* from )
- void **Nil** ( *AABB* from )

- void **Nil** (*Matrix3* from )
- void **Nil** (*Transform* from )
- void **Nil** (*Color* from )
- void **Nil** (*Image* from )
- void **Nil** (*NodePath* from )
- void **Nil** (*RID* from )
- void **Nil** (*Object* from )
- void **Nil** (*InputEvent* from )
- void **Nil** (*Dictionary* from )
- void **Nil** (*Array* from )
- void **Nil** (*RawArray* from )
- void **Nil** (*IntArray* from )
- void **Nil** (*RealArray* from )
- void **Nil** (*StringArray* from )
- void **Nil** (*Vector2Array* from )
- void **Nil** (*Vector3Array* from )
- void **Nil** (*ColorArray* from )

## 9.167 Node

**Inherits:** *Object*

**Inherited By:** *Viewport, Timer, CanvasLayer, EventPlayer, SoundRoomParams, Spatial, AnimationPlayer, EditorPlugin, ResourcePreloader, AnimationTreePlayer, SamplePlayer, InstancePlaceholder, StreamPlayer, CanvasItem, Tween*

**Category:** Core

### 9.167.1 Brief Description

Base class for all the “Scene” elements.

### 9.167.2 Member Functions

void	<code>_enter_tree ()</code> virtual
void	<code>_exit_tree ()</code> virtual
void	<code>_fixed_process (float delta)</code> virtual
void	<code>_input (InputEvent event)</code> virtual
void	<code>_process (float delta)</code> virtual
void	<code>_ready ()</code> virtual
void	<code>_unhandled_input (InputEvent event)</code> virtual

Continued on next page

Table 9.15 – continued from previous page

void	<code>_unhandled_key_input ( InputEvent key_event ) virtual</code>
void	<code>set_name ( String name )</code>
<i>String</i>	<code>get_name () const</code>
void	<code>add_child ( Node node, bool legible_unique_name=false )</code>
void	<code>remove_child ( Node node )</code>
<i>int</i>	<code>get_child_count () const</code>
<i>Array</i>	<code>get_children () const</code>
<i>Node</i>	<code>get_child ( int idx ) const</code>
<i>bool</i>	<code>has_node ( NodePath path ) const</code>
<i>Node</i>	<code>get_node ( NodePath path ) const</code>
<i>Parent</i>	<code>get_parent () const</code>
<i>Node</i>	<code>find_node ( String mask, bool recursive=true, bool owned=true ) const</code>
<i>bool</i>	<code>has_node_and_resource ( NodePath path ) const</code>
<i>Array</i>	<code>get_node_and_resource ( NodePath path )</code>
<i>bool</i>	<code>is_inside_tree () const</code>
<i>bool</i>	<code>is_a_parent_of ( Node node ) const</code>
<i>bool</i>	<code>is_greater_than ( Node node ) const</code>
<i>NodePath</i>	<code>get_path () const</code>
<i>NodePath</i>	<code>get_path_to ( Node node ) const</code>
void	<code>add_to_group ( String group, bool persistent=false )</code>
void	<code>remove_from_group ( String group )</code>
<i>bool</i>	<code>is_in_group ( String group ) const</code>
void	<code>move_child ( Node child_node, int to_pos )</code>
<i>Array</i>	<code>get_groups () const</code>
void	<code>raise ()</code>
void	<code>set_owner ( Node owner )</code>
<i>Node</i>	<code>get_owner () const</code>
void	<code>remove_and_skip ()</code>
<i>int</i>	<code>get_index () const</code>
void	<code>print_tree ()</code>
void	<code>set_filename ( String filename )</code>
<i>String</i>	<code>get_filename () const</code>
void	<code>propagate_notification ( int what )</code>
void	<code>set_fixed_process ( bool enable )</code>
<i>float</i>	<code>get_fixed_process_delta_time () const</code>
<i>bool</i>	<code>is_fixed_processing () const</code>
void	<code>set_process ( bool enable )</code>
<i>float</i>	<code>get_process_delta_time () const</code>
<i>bool</i>	<code>is_processing () const</code>
void	<code>set_process_input ( bool enable )</code>
<i>bool</i>	<code>is_processing_input () const</code>
void	<code>set_processUnhandledInput ( bool enable )</code>
<i>bool</i>	<code>is_processingUnhandledInput () const</code>
void	<code>set_processUnhandledKeyInput ( bool enable )</code>
<i>bool</i>	<code>is_processingUnhandledKeyInput () const</code>
void	<code>set_pause_mode ( int mode )</code>
<i>int</i>	<code>get_pause_mode () const</code>
<i>bool</i>	<code>can_process () const</code>
void	<code>print_stray_nodes ()</code>
<i>int</i>	<code>get_position_in_parent () const</code>

Continued on next page

Table 9.15 – continued from previous page

<i>SceneTree</i>	<code>get_tree() const</code>
<i>Node</i>	<code>duplicate( bool use_instancing=false ) const</code>
<i>void</i>	<code>replace_by( Node node, bool keep_data=false )</code>
<i>void</i>	<code>set_scene_instance_load_placeholder( bool load_placeholder )</code>
<i>bool</i>	<code>get_scene_instance_load_placeholder() const</code>
<i>Object</i>	<code>get_viewport() const</code>
<i>void</i>	<code>queue_free()</code>

### 9.167.3 Signals

- `renamed()`
- `enter_tree()`
- `exit_tree()`

### 9.167.4 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**
- **PAUSE\_MODE\_INHERIT = 0**
- **PAUSE\_MODE\_STOP = 1**
- **PAUSE\_MODE\_PROCESS = 2**

### 9.167.5 Description

Nodes can be set as children of other nodes, resulting in a tree arrangement. Any tree of nodes is called a “Scene”.

Scenes can be saved to disk, and then instanced into other scenes. This allows for very high flexibility in the architecture and data model of the projects.

SceneMainLoop contains the “active” tree of nodes, and a node becomes active (receiving NOTIFICATION\_ENTER\_SCENE) when added to that tree.

A node can contain any number of nodes as children (but there is only one tree root) with the requirement that no two children with the same name can exist.

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).

Nodes can be set to “process” state, so they constantly receive a callback requesting them to process (do anything). Normal processing (`_process`) happens as fast as possible and is dependent on the frame rate, so the processing time delta is variable. Fixed processing (`_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 with every input that the program receives. Since this is usually too overkill (unless used for simple projects), an `_unhandled_input` function is called when the input was not handled by anyone else (usually, GUI `Control` nodes).

To keep track of the scene hierarchy (specially when instancing scenes into scenes) an “owner” can be set to a node. This keeps track of who instanced what. This is mostly useful when writing editors and tools, though.

Finally, when a node is freed, it will free all its children nodes too.

## 9.167.6 Member Function Description

- `void _enter_tree () virtual`
- `void _exit_tree () virtual`
- `void _fixed_process (float delta) virtual`

Called for fixed processing (synced to the physics).

- `void _input (InputEvent event) virtual`

Called when any input happens (also must enable with `set_process_input` or the property).

- `void _process (float delta) virtual`

Called for processing. This is called every frame, with the delta time from the previous frame.

- `void _ready () virtual`

Called when ready (entered scene and children entered too).

- `void _unhandled_input (InputEvent event) virtual`

Called when any input happens that was not handled by something else (also must enable with `set_processUnhandledInput` or the property).

- `void _unhandled_key_input (InputEvent key_event) virtual`

Called when any key input happens that was not handled by something else.

- `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.

- `String get_name () const`

Return the name of the `Node`. Name is be unique within parent.

- `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 node with human-readable names, based on the name of node being instanced instead of its type only.

- `void remove_child ( Node node )`

Remove a child `Node`. Node is NOT deleted and will have to be deleted manually.

- `int get_child_count ( ) const`

Return the amount of children nodes.

- `Array get_children ( ) const`
- `Node get_child ( int idx ) const`

Return a children node by it's index (see `get_child_count`). This method is often used for iterating all children of a node.

- `bool has_node ( NodePath path ) const`
- `Node get_node ( NodePath path ) const`

Fetch a node. NodePath must be valid (or else error will occur) 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_scene`). Examples. Assume your current node is Character and 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")`
- Parent `get_parent ( ) const`

Return the parent `Node` of the current `Node`, or an empty Object if the node lacks a parent.

- `Node find_node ( String mask, bool recursive=true, bool owned=true ) const`
- `bool has_node_and_resource ( NodePath path ) const`
- `Array get_node_and_resource ( NodePath path )`
- `bool is_inside_tree ( ) 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_greater_than ( Node node ) const`

Return *true* if “node” occurs later in the scene hierarchy than the current node, otherwise return *false*.

- `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_scene`).

- `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.

- `void add_to_group ( String group, bool persistent=false )`

Add a node to a group. Groups are helpers to name and organize group of nodes, like for example: “Enemies”, “Collectables”, etc. A `Node` can be in any number of groups. Nodes can be assigned a group at any time, but will not be added to it until they are inside the scene tree (see `is_inside_scene`).

- `void remove_from_group ( String group )`

Remove a node from a group.

- `bool is_in_group ( String group ) 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.

- `Array get_groups ( ) const`
- `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 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.

- `Node get_owner ( ) const`

Get the node owner (see `set_node_owner`).

- `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.

- `int get_index ( ) const`

Get the node index in the parent (assuming it has a parent).

- `void print_tree ( )`

Print the scene to stdout. Used mainly for debugging purposes.

- `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, its topmost node contains the filename from where it was loaded.

- `String get_filename ( ) const`

Return a filename that may be containedA node can contain by the node. When a scene is instanced from a file, its topmost node contains the filename from where it was loaded (see [set\\_filename](#)).

- `void propagate_notification ( int what )`

Notify the current node and all its children recursively by calling notification() in all of them.

- `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\\_time](#).

- `float get_fixed_process_delta_time () const`

Return the time elapsed since the last fixed frame. This is always the same in fixed processing unless the frames per second is changed in [OS](#).

- `bool is_fixed_processing () const`

Return true if fixed processing is enabled (see [set\\_fixed\\_process](#)).

- `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\\_time](#).

- `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 is_processing () const`

Return whether processing is enabled in the current node (see [set\\_process](#)).

- `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](#)).

- `bool is_processing_input () const`

Return true if the node is processing input (see [set\\_process\\_input](#)).

- `void set_processUnhandledInput ( 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](#)).

- `bool is_processingUnhandledInput () const`

Return true if the node is processing unhandled input (see [set\\_processUnhandledInput](#)).

- `void setProcessUnhandledKeyInput ( bool enable )`

- `bool is_ProcessingUnhandledKeyInput () const`

- `void set_pause_mode ( int mode )`

- `int get_pause_mode () const`

- `bool can_process () const`

Return true if the node can process.

- `void printStrayNodes ()`

- `int get_position_in_parent() const`
- `SceneTree get_tree() const`
- `Node duplicate(bool use_instancing=false) const`
- `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_scene_instance_load_placeholder(bool load_placeholder)`
- `bool get_scene_instance_load_placeholder() const`
- `Object get_viewport() const`
- `void queue_free()`

## 9.168 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

### 9.168.1 Brief Description

Base node for 2D system.

## 9.168.2 Member Functions

void	<i>set_pos</i> ( <i>Vector2</i> pos )
void	<i>set_rot</i> ( <i>float</i> rot )
void	<i>set_scale</i> ( <i>Vector2</i> scale )
<i>Vector2</i>	<i>get_pos</i> ( ) const
<i>float</i>	<i>get_rot</i> ( ) const
<i>Vector2</i>	<i>get_scale</i> ( ) const
void	<i>rotate</i> ( <i>float</i> radians )
void	<i>move_local_x</i> ( <i>float</i> delta, <i>bool</i> scaled=false )
void	<i>move_local_y</i> ( <i>float</i> delta, <i>bool</i> scaled=false )
void	<i>translate</i> ( <i>Vector2</i> offset )
void	<i>global_translate</i> ( <i>Vector2</i> offset )
void	<i>scale</i> ( <i>Vector2</i> ratio )
void	<i>set_global_pos</i> ( <i>Vector2</i> pos )
<i>Vector2</i>	<i>get_global_pos</i> ( ) const
void	<i>set_transform</i> ( <i>Matrix32</i> xform )
void	<i>set_global_transform</i> ( <i>Matrix32</i> xform )
void	<i>look_at</i> ( <i>Vector2</i> point )
<i>float</i>	<i>get_angle_to</i> ( <i>Vector2</i> point ) const
void	<i>set_z</i> ( <i>int</i> z )
<i>int</i>	<i>get_z</i> ( ) const
void	<i>set_z_as_relative</i> ( <i>bool</i> enable )
<i>bool</i>	<i>is_z_relative</i> ( ) const
void	<i>edit_set_pivot</i> ( <i>Vector2</i> pivot )
<i>Matrix32</i>	<i>get_relative_transform_to_parent</i> ( <i>Object</i> parent ) const

## 9.168.3 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.

## 9.168.4 Member Function Description

- void **set\_pos** ( *Vector2* pos )

Set the position of the 2d node.

- void **set\_rot** ( *float* rot )

Set the rotation of the 2d node.

- void **set\_scale** ( *Vector2* scale )

Set the scale of the 2d node.

- *Vector2* **get\_pos** ( ) const

Return the position of the 2D node.

- *float* **get\_rot** ( ) const

Return the rotation of the 2D node.

- *Vector2* **get\_scale** ( ) const

Return the scale of the 2D node.

- void **rotate** (*float* radians)
- void **move\_local\_x** (*float* delta, *bool* scaled=false)
- void **move\_local\_y** (*float* delta, *bool* scaled=false)
- void **translate** (*Vector2* offset)
- void **global\_translate** (*Vector2* offset)
- void **scale** (*Vector2* ratio)
- void **set\_global\_pos** (*Vector2* pos)
- *Vector2* **get\_global\_pos** () const

Return the global position of the 2D node.

- void **set\_transform** (*Matrix32* xform)
- void **set\_global\_transform** (*Matrix32* xform)
- void **look\_at** (*Vector2* point)
- *float* **get\_angle\_to** (*Vector2* point) const
- void **set\_z** (*int* z)
- *int* **get\_z** () const
- void **set\_z\_as\_relative** (*bool* enable)
- *bool* **is\_z\_relative** () const
- void **edit\_set\_pivot** (*Vector2* pivot)
- *Matrix32* **get\_relative\_transform\_to\_parent** (*Object* parent) const

## 9.169 NodePath

**Category:** Built-In Types

### 9.169.1 Brief Description

Built-in type optimized for path traversing.

### 9.169.2 Member Functions

<i>String</i>	<i>get_name</i> ( <i>int</i> idx)
<i>int</i>	<i>get_name_count</i> ()
<i>String</i>	<i>get_property</i> ()
<i>String</i>	<i>get_subname</i> ( <i>int</i> idx)
<i>int</i>	<i>get_subname_count</i> ()
<i>bool</i>	<i>is_absolute</i> ()
<i>bool</i>	<i>is_empty</i> ()
<i>NodePath</i>	<i>NodePath</i> ( <i>String</i> from)

### 9.169.3 Description

Built-in type optimized for path traversing. A Node path is an optimized compiled path used for traversing the scene tree. It references nodes and can reference properties in that node, or even reference properties inside the resources of the node.

### 9.169.4 Member Function Description

- `String get_name ( int idx )`

Return a path level name.

- `int get_name_count ()`

Return the path level count.

- `String get_property ()`

Return the property associated (empty if none).

- `String get_subname ( int idx )`

Return the subname level name.

- `int get_subname_count ()`

Return the subname count.

- `bool is_absolute ()`

Return true if the node path is absolute (not relative).

- `bool is_empty ()`

Return true if the node path is empty.

- `NodePath NodePath ( String from )`

## 9.170 Object

**Inherited By:** *Reference, Physics2DServer, Input, SpatialSound2DServer, Node, Geometry, TreeItem, Physics-DirectSpaceState, Physics2DDirectSpaceState, MainLoop, InputMap, UndoRedo, PhysicsServer, Resource-Saver, Performance, PathRemap, ResourceLoader, AudioServer, SpatialSoundServer, IP, VisualServer, OS, Physics2DDirectBodyState, Globals, PhysicsDirectBodyState, TranslationServer*

**Category:** Core

### 9.170.1 Brief Description

Base class for all non built-in types.

### 9.170.2 Member Functions

void	<code>_get ( String property ) virtual</code>
<code>Array</code>	<code>_get_property_list () virtual</code>

Table 9.16 – continued from previous page

void	<code>_init () virtual</code>
void	<code>_notification ( int what ) virtual</code>
void	<code>_set ( String property, var value ) virtual</code>
void	<code>free ()</code>
<code>String</code>	<code>get_type () const</code>
<code>bool</code>	<code>is_type ( String type ) const</code>
void	<code>set ( String property, var value )</code>
void	<code>get ( String property ) const</code>
<code>Array</code>	<code>get_property_list () const</code>
<code>Array</code>	<code>get_method_list () const</code>
void	<code>notification ( int what, bool reversed=false )</code>
<code>int</code>	<code>get_instance_ID () const</code>
void	<code>set_script ( Script script )</code>
<code>Script</code>	<code>get_script () const</code>
void	<code>set_meta ( String name, var value )</code>
void	<code>get_meta ( String name ) const</code>
<code>bool</code>	<code>has_meta ( String name ) const</code>
<code>StringArray</code>	<code>get_meta_list () const</code>
void	<code>add_user_signal ( String signal, Array arguments=Array() )</code>
<code>bool</code>	<code>has_user_signal ( String signal ) const</code>
void	<code>emit_signal ( String signal, var arg0=NULL, var arg1=NULL, var arg2=NULL, var arg3=NULL, var arg4=NULL )</code>
void	<code>call ( String method, var arg0=NULL, var arg1=NULL, var arg2=NULL, var arg3=NULL, var arg4=NULL, var arg5=NULL )</code>
void	<code>call_deferred ( String method, var arg0=NULL, var arg1=NULL, var arg2=NULL, var arg3=NULL, var arg4=NULL )</code>
Variant	<code>callv ( String method, Array arg_array )</code>
<code>bool</code>	<code>has_method ( String method ) const</code>
<code>Array</code>	<code>get_signal_list () const</code>
<code>Array</code>	<code>get_signal_connection_list ( String signal ) const</code>
<code>int</code>	<code>connect ( String signal, Object target, String method, Array binds=Array(), int flags=0 )</code>
void	<code>disconnect ( String signal, Object target, String method )</code>
<code>bool</code>	<code>is_connected ( String signal, Object target, String method ) const</code>
void	<code>set_block_signals ( bool enable )</code>
<code>bool</code>	<code>is_blocking_signals () const</code>
void	<code>set_message_translation ( bool enable )</code>
<code>bool</code>	<code>can_translate_messages () const</code>
void	<code>property_list_changed_notify ()</code>
<code>String</code>	<code>XL_MESSAGE ( String message ) const</code>
<code>String</code>	<code>tr ( String message ) const</code>
<code>bool</code>	<code>is_queued_for_deletion () const</code>

### 9.170.3 Signals

- `script_changed ()`

### 9.170.4 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.

## 9.170.5 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 derivates 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.

## 9.170.6 Member Function Description

- `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, var value ) virtual`

Set a property. Return true if the property was found.

- `void free ()`

- `String get_type () const`

Return the type of the object as a string.

- `bool is_type ( String type ) const`

Check the type of the object against a string (including inheritance).

- `void set ( String property, var value )`

Set property into the object.

- `void get ( String property ) const`

Get a property from the object.

- `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).

- `Array get_method_list () const`
- `void notification ( int what, bool reversed=false )`

Notify the object of something.

- `int get_instance_ID () const`

Return the instance ID. All objects have a unique instance ID.

- `void set_script ( Script script )`

Set a script into the object, scripts extend the object functionality.

- `Script get_script () const`

Return the object script (or null if it doesn't have one).

- `void set_meta ( String name, var value )`

Set a metadata into the object. Metadata is serialized. Metadata can be *anything*.

- `void get_meta ( String name ) const`

Return a metadata from the object.

- `bool has_meta ( String name ) const`

Return true if a metadata is found with the requested name.

- `StringArray get_meta_list () const`

Return the list of metadata in the object.

- `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\_\*).

- `bool has_user_signal ( String signal ) const`
- `void emit_signal ( String signal, var arg0=NULL, var arg1=NULL, var arg2=NULL, var arg3=NULL, var arg4=NULL )`

Emit a signal. Arguments are passed in an array.

- `void call ( String method, var arg0=NULL, var arg1=NULL, var arg2=NULL, var arg3=NULL, var arg4=NULL, var arg5=NULL, var arg6=NULL, var arg7=NULL, var arg8=NULL, var arg9=NULL )`

Call a function in the object, result is returned.

- `void call_deferred ( String method, var arg0=NULL, var arg1=NULL, var arg2=NULL, var arg3=NULL, var 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 has_method ( String method ) const`
- `Array get_signal_list () const`

Return the list of signals as an array of dictionaries.

- `Array get_signal_connection_list ( String signal ) const`
- `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.

- `bool is_connected ( String signal, Object target, String method ) const`

Return true if a connection exists for a given signal and target/method.

- `void set_block_signals ( bool enable )`

If set to true, signal emission is blocked.

- `bool is_blocking_signals ( ) const`

Return true if signal emission blocking is enabled.

- `void set_message_translation ( bool enable )`

Set true if this object can translate strings (in calls to `tr()`). Default is true.

- `bool can_translate_messages ( ) const`

Return true if this object can translate strings.

- `void property_list_changed_notify ( )`

- `String XL_MESSAGE ( String message ) const`

Deprecated, will go away.

- `String tr ( String message ) const`

Translate a message. Only works in message translation is enabled (which is by default). See `set_message_translation`.

- `bool is_queued_for_deletion ( ) const`

## 9.171 OccluderPolygon2D

**Inherits:** `Resource < Reference < Object`

**Category:** Core

### 9.171.1 Brief Description

### 9.171.2 Member Functions

void	<code>set_closed ( bool closed )</code>
<i>bool</i>	<code>is_closed () const</code>
void	<code>set_cull_mode ( int cull_mode )</code>
<i>int</i>	<code>get_cull_mode () const</code>
void	<code>set_polygon ( Vector2Array polygon )</code>
<i>Vector2Array</i>	<code>get_polygon () const</code>

### 9.171.3 Numeric Constants

- **CULL\_DISABLED = 0**
- **CULL\_CLOCKWISE = 1**
- **CULL\_COUNTER\_CLOCKWISE = 2**

### 9.171.4 Member Function Description

- void `set_closed ( bool closed )`
- *bool* `is_closed () const`
- void `set_cull_mode ( int cull_mode )`
- *int* `get_cull_mode () const`
- void `set_polygon ( Vector2Array polygon )`
- *Vector2Array* `get_polygon () const`

## 9.172 OmniLight

**Inherits:** [Light](#) < [VisualInstance](#) < [Spatial](#) < [Node](#) < [Object](#)

**Category:** Core

### 9.172.1 Brief Description

OmniDirectional Light, such as a light bulb or a candle.

### 9.172.2 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.

## 9.173 OptionButton

Inherits: [Button](#) < [BaseButton](#) < [Control](#) < [CanvasItem](#) < [Node](#) < [Object](#)

Category: Core

### 9.173.1 Brief Description

Button control that provides selectable options when pressed.

### 9.173.2 Member Functions

void	<a href="#">add_item</a> ( <i>String</i> label, <i>int</i> id=-1 )
void	<a href="#">add_icon_item</a> ( <i>Texture</i> texture, <i>String</i> label, <i>int</i> id )
void	<a href="#">set_item_text</a> ( <i>int</i> idx, <i>String</i> text )
void	<a href="#">set_item_icon</a> ( <i>int</i> idx, <i>Texture</i> texture )
void	<a href="#">set_item_disabled</a> ( <i>int</i> idx, <i>bool</i> disabled )
void	<a href="#">set_item_ID</a> ( <i>int</i> idx, <i>int</i> id )
void	<a href="#">set_item_metadata</a> ( <i>int</i> idx, var metadata )
<i>String</i>	<a href="#">get_item_text</a> ( <i>int</i> idx ) const
<i>Texture</i>	<a href="#">get_item_icon</a> ( <i>int</i> idx ) const
<i>int</i>	<a href="#">get_item_ID</a> ( <i>int</i> idx ) const
void	<a href="#">get_item_metadata</a> ( <i>int</i> idx ) const
<i>bool</i>	<a href="#">is_item_disabled</a> ( <i>int</i> idx ) const
<i>int</i>	<a href="#">get_item_count</a> ( ) const
void	<a href="#">add_separator</a> ( )
void	<a href="#">clear</a> ( )
void	<a href="#">select</a> ( <i>int</i> idx )
<i>int</i>	<a href="#">get_selected</a> ( ) const
<i>int</i>	<a href="#">get_selected_ID</a> ( ) const
void	<a href="#">get_selected_metadata</a> ( ) const
void	<a href="#">remove_item</a> ( <i>int</i> idx )

### 9.173.3 Signals

- [item\\_selected](#) ( *int* ID )

### 9.173.4 Description

OptionButton 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.

### 9.173.5 Member Function Description

- 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\\_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 set_item_text ( int idx, String text )`

Set the text of an item at index “idx”.

- `void set_item_icon ( int idx, Texture texture )`

Set the icon of an item at index “idx”.

- `void set_item_disabled ( int idx, bool disabled )`
- `void set_item_ID ( int idx, int id )`

Set the ID of an item at index “idx”.

- `void set_item_metadata ( int idx, var metadata )`
- `String get_item_text ( int idx ) const`

Return the text of the item at index “idx”.

- `Texture get_item_icon ( int idx ) const`

Return the icon of the item at index “idx”.

- `int get_item_ID ( int idx ) const`

Return the ID of the item at index “idx”.

- `void get_item_metadata ( int idx ) const`
- `bool is_item_disabled ( int idx ) const`
- `int get_item_count ( ) const`

Return the amount of items in the *OptionButton*.

- `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*.

- `void select ( int idx )`

Select an item by index and make it the current item.

- `int get_selected ( ) const`

Return the current item index

- `int get_selected_ID ( ) const`
- `void get_selected_metadata ( ) const`
- `void remove_item ( int idx )`

## 9.174 OS

**Inherits:** *Object*

**Category:** Core

### 9.174.1 Brief Description

Operating System functions.

### 9.174.2 Member Functions

void	<code>set_clipboard ( String clipboard )</code>
<code>String</code>	<code>get_clipboard () const</code>
void	<code>set_video_mode ( Vector2 size, bool fullscreen, bool resizable, int screen=0 )</code>
<code>Vector2</code>	<code>get_video_mode_size ( int screen=0 ) const</code>
<code>bool</code>	<code>is_video_mode_fullscreen ( int screen=0 ) const</code>
<code>bool</code>	<code>is_video_mode_resizable ( int screen=0 ) const</code>
<code>Array</code>	<code>get_fullscreen_mode_list ( int screen=0 ) const</code>
<code>int</code>	<code>get_screen_count () const</code>
<code>int</code>	<code>get_current_screen () const</code>
void	<code>set_current_screen ( int screen )</code>
<code>Vector2</code>	<code>get_screen_position ( int screen=0 ) const</code>
<code>Vector2</code>	<code>get_screen_size ( int screen=0 ) const</code>
<code>Vector2</code>	<code>get_window_position () const</code>
void	<code>set_window_position ( Vector2 position )</code>
<code>Vector2</code>	<code>get_window_size () const</code>
void	<code>set_window_size ( Vector2 size )</code>
void	<code>set_windowFullscreen ( bool enabled )</code>
<code>bool</code>	<code>is_windowFullscreen () const</code>
void	<code>set_windowResizable ( bool enabled )</code>
<code>bool</code>	<code>is_windowResizable () const</code>
void	<code>set_windowMinimized ( bool enabled )</code>
<code>bool</code>	<code>is_windowMinimized () const</code>
void	<code>set_windowMaximized ( bool enabled )</code>
<code>bool</code>	<code>is_windowMaximized () const</code>
void	<code>set_screen_orientation ( int orientation )</code>
<code>int</code>	<code>get_screen_orientation () const</code>
void	<code>set_keep_screen_on ( bool enabled )</code>
<code>bool</code>	<code>is_keep_screen_on () const</code>
void	<code>set_iterations_per_second ( int iterations_per_second )</code>
<code>int</code>	<code>get_iterations_per_second () const</code>
void	<code>set_target_fps ( int target_fps )</code>
<code>float</code>	<code>get_target_fps () const</code>
void	<code>set_time_scale ( float time_scale )</code>
<code>float</code>	<code>get_time_scale ()</code>
<code>bool</code>	<code>has_touchscreen_ui_hint () const</code>
void	<code>set_window_title ( String title )</code>
void	<code>set_low_processor_usage_mode ( bool enable )</code>
<code>bool</code>	<code>is_in_low_processor_usage_mode () const</code>
<code>int</code>	<code>get_processor_count () const</code>
<code>String</code>	<code>get_executable_path () const</code>
<code>int</code>	<code>execute ( String path, StringArray arguments, bool blocking, Array output=Array() )</code>
<code>int</code>	<code>kill ( int pid )</code>
<code>int</code>	<code>shell_open ( String uri )</code>
<code>int</code>	<code>get_process_ID () const</code>

Continued on next page

Table 9.17 – continued from previous page

<i>String</i>	<code>get_environment ( String environment ) const</code>
<i>bool</i>	<code>has_environment ( String environment ) const</code>
<i>String</i>	<code>get_name () const</code>
<i>StringArray</i>	<code>get_cmdline_args ()</code>
<i>Object</i>	<code>get_main_loop () const</code>
<i>Dictionary</i>	<code>get_date ( bool utc=false ) const</code>
<i>Dictionary</i>	<code>get_time ( bool utc=false ) const</code>
<i>Dictionary</i>	<code>get_time_zone_info () const</code>
<i>int</i>	<code>get_unix_time () const</code>
<i>int</i>	<code>get_system_time_secs () const</code>
<i>void</i>	<code>set_icon ( Image icon )</code>
<i>void</i>	<code>delay_usec ( int usec ) const</code>
<i>void</i>	<code>delay_msec ( int msec ) const</code>
<i>int</i>	<code>get_ticks_msec () const</code>
<i>int</i>	<code>get_splash_tick_msec () const</code>
<i>String</i>	<code>get_locale () const</code>
<i>String</i>	<code>get_model_name () const</code>
<i>String</i>	<code>get_custom_level () const</code>
<i>bool</i>	<code>can_draw () const</code>
<i>int</i>	<code>get_frames_drawn ()</code>
<i>bool</i>	<code>is_stdout_verbose () const</code>
<i>bool</i>	<code>can_use_threads () const</code>
<i>bool</i>	<code>is_debug_build () const</code>
<i>void</i>	<code>dump_memory_to_file ( String file )</code>
<i>void</i>	<code>dump_resources_to_file ( String file )</code>
<i>void</i>	<code>print_resources_in_use ( bool short=false )</code>
<i>void</i>	<code>print_all_resources ( String tofile="" )</code>
<i>int</i>	<code>get_static_memory_usage () const</code>
<i>int</i>	<code>get_static_memory_peak_usage () const</code>
<i>int</i>	<code>get_dynamic_memory_usage () const</code>
<i>String</i>	<code>get_data_dir () const</code>
<i>String</i>	<code>get_system_dir ( int dir ) const</code>
<i>String</i>	<code>get_unique_ID () const</code>
<i>bool</i>	<code>is_ok_left_and_cancel_right () const</code>
<i>float</i>	<code>get_frames_per_second () const</code>
<i>void</i>	<code>print_all_textures_by_size ()</code>
<i>void</i>	<code>print_resources_by_type ( StringArray types )</code>
<i>int</i>	<code>native_video_play ( String path, float volume, String audio_track, String subtitle_track )</code>
<i>bool</i>	<code>native_video_is_playing ()</code>
<i>void</i>	<code>native_video_stop ()</code>
<i>void</i>	<code>native_video_pause ()</code>
<i>String</i>	<code>get_scancode_string ( int code ) const</code>
<i>bool</i>	<code>is_scancode_unicode ( int code ) const</code>
<i>int</i>	<code>find_scancode_from_string ( String string ) const</code>
<i>void</i>	<code>set_use_file_access_save_and_swap ( bool enabled )</code>
<i>void</i>	<code>alert ( String text, String title="Alert!" )</code>
<i>int</i>	<code>set_thread_name ( String name )</code>

### 9.174.3 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** = 0
- **MONTH\_FEBRUARY** = 1
- **MONTH\_MARCH** = 2
- **MONTH\_APRIIL** = 3
- **MONTH\_MAY** = 4
- **MONTH\_JUNE** = 5
- **MONTH\_JULY** = 6
- **MONTH\_AUGUST** = 7
- **MONTH\_SEPTEMBER** = 8
- **MONTH\_OCTOBER** = 9
- **MONTH\_NOVEMBER** = 10
- **MONTH\_DECEMBER** = 11
- **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

## 9.174.4 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 / Time
- Timers
- Environment Variables
- Execution of Binaries
- Command Line

## 9.174.5 Member Function Description

- `void set_clipboard ( String clipboard )`

Set clipboard to the OS.

- `String get_clipboard () const`

Get clipboard from the host OS.

- `void set_video_mode ( Vector2 size, bool fullscreen, bool resizable, int screen=0 )`

Change the video mode.

- `Vector2 get_video_mode_size ( int screen=0 ) const`

Return the current video mode size.

- `bool is_video_modeFullscreen ( int screen=0 ) const`

Return true if the current video mode is fullscreen.

- `bool is_video_modeResizable ( int screen=0 ) const`

Return true if the window is resizable.

- `Array get_fullscreen_mode_list ( int screen=0 ) const`

Return the list of fullscreen modes.

- `int get_screen_count () const`

- `int get_current_screen () const`

- `void set_current_screen ( int screen )`

- `Vector2 get_screen_position ( int screen=0 ) const`

- `Vector2 get_screen_size ( int screen=0 ) const`

- `Vector2 get_window_position () const`

- `void set_window_position ( Vector2 position )`

- `Vector2 get_window_size () const`

- void **set\_window\_size** ( *Vector2* size )
- void **set\_window\_fullscreen** ( *bool* enabled )
- *bool* **is\_windowFullscreen** ( ) const
- void **set\_window\_resizable** ( *bool* enabled )
- *bool* **is\_windowResizable** ( ) const
- void **set\_window\_minimized** ( *bool* enabled )
- *bool* **is\_windowMinimized** ( ) const
- void **set\_window\_maximized** ( *bool* enabled )
- *bool* **is\_windowMaximized** ( ) const
- void **set\_screen\_orientation** ( *int* orientation )
- *int* **get\_screen\_orientation** ( ) const
- 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)

- *bool* **is\_keep\_screen\_on** ( ) const
- void **set\_iterations\_per\_second** ( *int* iterations\_per\_second )

Set the amount of fixed iterations per second (for fixed process and physics).

- *int* **get\_iterations\_per\_second** ( ) const

Return the amount of fixed iterations per second (for fixed process and physics).

- void **set\_target\_fps** ( *int* target\_fps )
- *float* **get\_target\_fps** ( ) const
- void **set\_time\_scale** ( *float* time\_scale )
- *float* **get\_time\_scale** ( )
- *bool* **has\_touchscreen\_ui\_hint** ( ) const
- void **set\_window\_title** ( *String* title )
- 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.

- *bool* **is\_in\_low\_processor\_usage\_mode** ( ) const

Return true if low cpu usage mode is enabled.

- *int* **get\_processor\_count** ( ) const
- *String* **get\_executable\_path** ( ) const

Return the path to the current engine executable.

- *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* **kill** ( *int* pid )

Kill a process ID.

- `int shell_open ( String uri )`
- `int get_process_ID () const`
- `String get_environment ( String environment ) const`

Return an environment variable.

- `bool has_environment ( String environment ) const`

Return true if an environment variable exists.

- `String get_name () const`

Return the name of the host OS.

- `StringArray get_cmdline_args ()`

Return the commandline passed to the engine.

- `Object get_main_loop () const`

Return the main loop object (see [MainLoop](#)).

- `Dictionary get_date ( bool utc=false ) const`
- `Dictionary get_time ( bool utc=false ) const`
- `Dictionary get_time_zone_info () const`
- `int get_unix_time () const`
- `int get_system_time_secs () const`
- `void set_icon ( Image icon )`
- `void delay_usecs ( int usec ) const`

Delay executing of the current thread by given microseconds.

- `void delay_msec ( int msec ) const`

Delay executing of the current thread by given milliseconds.

- `int get_ticks_msec () const`

Return the amount of time passed in milliseconds since the engine started.

- `int get_splash_tick_msec () const`
- `String get_locale () const`

Return the host OS locale.

- `String get_model_name () const`
- `String get_custom_level () const`
- `bool can_draw () const`

Return true if the host OS allows drawing.

- `int get_frames_drawn ()`

Return the total amount of frames drawn.

- `bool is_stdout_verbose () const`

Return true if the engine was executed with -v (verbose stdout).

- `bool can_use_threads () const`

- `bool is_debug_build () const`
- `void dump_memory_to_file ( String file )`
- `void dump_resources_to_file ( String file )`
- `void print_resources_in_use ( bool short=false )`
- `void print_all_resources ( String tofile="" )`
- `int get_static_memory_usage () const`
- `int get_static_memory_peak_usage () const`

Return the max amount of static memory used (only works in debug).

- `int get_dynamic_memory_usage () const`

Return the total amount of dynamic memory used (only works in debug).

- `String get_data_dir () const`
- `String get_system_dir ( int dir ) const`
- `String get_unique_ID () const`
- `bool is_ok_left_and_cancel_right () const`
- `float get_frames_per_second () const`
- `void print_all_textures_by_size ()`
- `void print_resources_by_type ( StringArray types )`
- `int native_video_play ( String path, float volume, String audio_track, String subtitle_track )`
- `bool native_video_is_playing ()`
- `void native_video_stop ()`
- `void native_video_pause ()`
- `String get_scancode_string ( int code ) const`
- `bool is_scancode_unicode ( int code ) const`
- `int find_scancode_from_string ( String string ) const`
- `void set_use_file_access_save_and_swap ( bool enabled )`
- `void alert ( String text, String title="Alert!" )`
- `int set_thread_name ( String name )`

## 9.175 PackedDataContainer

Inherits: [Resource](#) < [Reference](#) < [Object](#)

Category: Core

### 9.175.1 Brief Description

### 9.175.2 Member Functions

Error	<code>pack ( var value )</code>
<code>int</code>	<code>size () const</code>

### 9.175.3 Member Function Description

- Error `pack ( var value )`
- `int size () const`

## 9.176 PackedDataContainerRef

**Inherits:** *Reference < Object*

**Category:** Core

### 9.176.1 Brief Description

### 9.176.2 Member Functions

<code>int</code>	<code>size () const</code>
------------------	----------------------------

### 9.176.3 Member Function Description

- `int size () const`

## 9.177 PackedScene

**Inherits:** *Resource < Reference < Object*

**Category:** Core

### 9.177.1 Brief Description

### 9.177.2 Member Functions

<code>int</code>	<code>pack ( Node path )</code>
<code>Node</code>	<code>instance ( bool gen_edit_state=false ) const</code>
<code>bool</code>	<code>can_instance () const</code>
<code>SceneState</code>	<code>get_state ()</code>

### 9.177.3 Description

TODO: explain ownership, and that node does not need to own itself

## 9.177.4 Member Function Description

- `int pack ( Node path )`

Pack will ignore any sub-nodes not owned by given node. See `Node.set_owner`.

- `Node instance ( bool gen_edit_state=false ) const`
- `bool can_instance ( ) const`
- `SceneState get_state ( )`

## 9.178 PacketPeer

**Inherits:** `Reference < Object`

**Inherited By:** `PacketPeerStream, PacketPeerUDP`

**Category:** Core

### 9.178.1 Brief Description

Abstraction and base class for packet-based protocols.

### 9.178.2 Member Functions

<code>void</code>	<code>get_var ( ) const</code>
<code>int</code>	<code>put_var ( Variant var )</code>
<code>RawArray</code>	<code>get_packet ( ) const</code>
<code>Error</code>	<code>put_packet ( RawArray buffer )</code>
<code>Error</code>	<code>get_packet_error ( ) const</code>
<code>int</code>	<code>get_available_packet_count ( ) const</code>

### 9.178.3 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.

### 9.178.4 Member Function Description

- `void get_var ( ) const`
- `int put_var ( Variant var )`
- `RawArray get_packet ( ) const`
- `Error put_packet ( RawArray buffer )`
- `Error get_packet_error ( ) const`
- `int get_available_packet_count ( ) const`

## 9.179 PacketPeerStream

**Inherits:** [PacketPeer](#) < [Reference](#) < [Object](#)

**Category:** Core

### 9.179.1 Brief Description

Wrapper to use a PacketPeer over a StreamPeer.

### 9.179.2 Member Functions

void	<code>set_stream_peer ( StreamPeer peer )</code>
------	--

### 9.179.3 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.

### 9.179.4 Member Function Description

- void `set_stream_peer ( StreamPeer peer )`

Set the StreamPeer object to be wrapped

## 9.180 PacketPeerUDP

**Inherits:** [PacketPeer](#) < [Reference](#) < [Object](#)

**Category:** Core

### 9.180.1 Brief Description

### 9.180.2 Member Functions

Error	<code>listen ( int port, int recv_buf_size=65536 )</code>
void	<code>close ()</code>
Error	<code>wait ()</code>
bool	<code>is_listening () const</code>
String	<code>get_packet_ip () const</code>
int	<code>get_packet_address () const</code>
int	<code>get_packet_port () const</code>
int	<code>set_send_address ( String host, int port )</code>

### 9.180.3 Member Function Description

- Error **listen** ( *int* port, *int* recv\_buf\_size=65536 )
- void **close** ()
- Error **wait** ()
- *bool* **is\_listening** () const
- *String* **get\_packet\_ip** () const
- *int* **get\_packet\_address** () const
- *int* **get\_packet\_port** () const
- *int* **set\_send\_address** ( *String* host, *int* port )

## 9.181 Panel

**Inherits:** *Control* < *CanvasItem* < *Node* < *Object*

**Category:** Core

### 9.181.1 Brief Description

Provides an opaque background for *Control* children.

### 9.181.2 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. image<class\_image>‘images/panel\_example.png:ref:/image<class\_/image>‘

## 9.182 PanelContainer

**Inherits:** *Container* < *Control* < *CanvasItem* < *Node* < *Object*

**Category:** Core

### 9.182.1 Brief Description

Panel container type.

### 9.182.2 Description

Panel container type. This container fits controls inside of the delimited area of a stylebox. It's useful for giving controls an outline.

## 9.183 ParallaxBackground

**Inherits:** [CanvasLayer](#) < [Node](#) < [Object](#)

**Category:** Core

### 9.183.1 Brief Description

### 9.183.2 Member Functions

void	<code>set_scroll_offset ( Vector2 ofs )</code>
<code>Vector2</code>	<code>get_scroll_offset () const</code>
void	<code>set_scroll_base_offset ( Vector2 ofs )</code>
<code>Vector2</code>	<code>get_scroll_base_offset () const</code>
void	<code>set_scroll_base_scale ( Vector2 scale )</code>
<code>Vector2</code>	<code>get_scroll_base_scale () const</code>
void	<code>set_limit_begin ( Vector2 ofs )</code>
<code>Vector2</code>	<code>get_limit_begin () const</code>
void	<code>set_limit_end ( Vector2 ofs )</code>
<code>Vector2</code>	<code>get_limit_end () const</code>
void	<code>set_ignore_camera_zoom ( bool ignore )</code>
<code>bool</code>	<code>is_ignore_camera_zoom ()</code>

### 9.183.3 Member Function Description

- void `set_scroll_offset ( Vector2 ofs )`
- `Vector2 get_scroll_offset () const`
- void `set_scroll_base_offset ( Vector2 ofs )`
- `Vector2 get_scroll_base_offset () const`
- void `set_scroll_base_scale ( Vector2 scale )`
- `Vector2 get_scroll_base_scale () const`
- void `set_limit_begin ( Vector2 ofs )`
- `Vector2 get_limit_begin () const`
- void `set_limit_end ( Vector2 ofs )`
- `Vector2 get_limit_end () const`
- void `set_ignore_camera_zoom ( bool ignore )`
- `bool is_ignore_camera_zoom ()`

## 9.184 ParallaxLayer

**Inherits:** [Node2D](#) < [CanvasItem](#) < [Node](#) < [Object](#)

**Category:** Core

### 9.184.1 Brief Description

### 9.184.2 Member Functions

void	<code>set_motion_scale ( Vector2 scale )</code>
<code>Vector2</code>	<code>get_motion_scale () const</code>
void	<code>set_mirroring ( Vector2 mirror )</code>
<code>Vector2</code>	<code>get_mirroring () const</code>

### 9.184.3 Member Function Description

- void `set_motion_scale ( Vector2 scale )`
- `Vector2 get_motion_scale () const`
- void `set_mirroring ( Vector2 mirror )`
- `Vector2 get_mirroring () const`

## 9.185 ParticleAttractor2D

Inherits: `Node2D < CanvasItem < Node < Object`

Category: Core

### 9.185.1 Brief Description

### 9.185.2 Member Functions

void	<code>set_enabled ( bool enabled )</code>
<code>bool</code>	<code>is_enabled () const</code>
void	<code>set_radius ( float radius )</code>
<code>float</code>	<code>get_radius () const</code>
void	<code>set_disable_radius ( float radius )</code>
<code>float</code>	<code>get_disable_radius () const</code>
void	<code>set_gravity ( float gravity )</code>
<code>float</code>	<code>get_gravity () const</code>
void	<code>set_absorption ( float absorption )</code>
<code>float</code>	<code>get_absorption () const</code>
void	<code>set_particles_path ( NodePath path )</code>
<code>NodePath</code>	<code>get_particles_path () const</code>

### 9.185.3 Member Function Description

- void `set_enabled ( bool enabled )`
- `bool is_enabled () const`
- void `set_radius ( float radius )`
- `float get_radius () const`

- void `set_disable_radius` (`float` radius)
- `float` `get_disable_radius` () const
- void `set_gravity` (`float` gravity)
- `float` `get_gravity` () const
- void `set_absorption` (`float` absorption)
- `float` `get_absorption` () const
- void `set_particles_path` (`NodePath` path)
- `NodePath` `get_particles_path` () const

## 9.186 Particles

**Inherits:** `GeometryInstance` < `VisualInstance` < `Spatial` < `Node` < `Object`

**Category:** Core

### 9.186.1 Brief Description

Particle system 3D Node

### 9.186.2 Member Functions

void	<code>set_amount</code> ( <code>int</code> amount)
<code>int</code>	<code>get_amount</code> () const
void	<code>set_emitting</code> ( <code>bool</code> enabled)
<code>bool</code>	<code>is_emitting</code> () const
void	<code>set_visibility_aabb</code> ( <code>AABB</code> aabb)
<code>AABB</code>	<code>get_visibility_aabb</code> () const
void	<code>set_emission_half_extents</code> ( <code>Vector3</code> half_extents)
<code>Vector3</code>	<code>get_emission_half_extents</code> () const
void	<code>set_emission_base_velocity</code> ( <code>Vector3</code> base_velocity)
<code>Vector3</code>	<code>get_emission_base_velocity</code> () const
void	<code>set_emission_points</code> ( <code>Vector3Array</code> points)
<code>Vector3Array</code>	<code>get_emission_points</code> () const
void	<code>set_gravity_normal</code> ( <code>Vector3</code> normal)
<code>Vector3</code>	<code>get_gravity_normal</code> () const
void	<code>set_variable</code> ( <code>int</code> variable, <code>float</code> value)
<code>float</code>	<code>get_variable</code> ( <code>int</code> variable) const
void	<code>set_randomness</code> ( <code>int</code> variable, <code>float</code> randomness)
<code>float</code>	<code>get_randomness</code> ( <code>int</code> variable) const
void	<code>set_color_phase_pos</code> ( <code>int</code> phase, <code>float</code> pos)
<code>float</code>	<code>get_color_phase_pos</code> ( <code>int</code> phase) const
void	<code>set_color_phase_color</code> ( <code>int</code> phase, <code>Color</code> color)
<code>Color</code>	<code>get_color_phase_color</code> ( <code>int</code> phase) const
void	<code>set_material</code> ( <code>Material</code> material)
<code>Material</code>	<code>get_material</code> () const

Continued on next page

Table 9.18 – continued from previous page

void	<code>set_emit_timeout ( float timeout )</code>
<i>float</i>	<code>get_emit_timeout () const</code>
void	<code>set_height_from_velocity ( bool enable )</code>
<i>bool</i>	<code>has_height_from_velocity () const</code>
void	<code>set_use_local_coordinates ( bool enable )</code>
<i>bool</i>	<code>is_using_local_coordinates () const</code>
void	<code>set_color_phases ( int count )</code>
<i>int</i>	<code>get_color_phases () const</code>

### 9.186.3 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**

### 9.186.4 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).

### 9.186.5 Member Function Description

- void **set\_amount** ( *int* amount )

Set total amount of particles in the system.

- *int* **get\_amount** ( ) const

Return the total amount of particles in the system.

- void **set\_emitting** ( *bool* enabled )

Set the “emitting” property state. When emitting, the particle system generates new particles at constant rate.

- *bool* **is\_emitting** ( ) const

Return the “emitting” property state (see `set_emitting`).

- `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.

- `AABB get_visibility_aabb () const`

Return the current visibility AABB.

- `void set_emission_half_extents (Vector3 half_extents)`

Set the half extents for the emission box.

- `Vector3 get_emission_half_extents () const`

Return the half extents for the emission box.

- `void set_emission_base_velocity (Vector3 base_velocity)`

- `Vector3 get_emission_base_velocity () const`

- `void set_emission_points (Vector3Array points)`

- `Vector3Array get_emission_points () const`

- `void set_gravity_normal (Vector3 normal)`

Set the normal vector towards where gravity is pulling (by default, negative Y).

- `Vector3 get_gravity_normal () const`

Return the normal vector towards where gravity is pulling (by default, negative Y).

- `void set_variable (int variable, float value)`

Set a specific variable for the particle system (see `VAR_*` enum).

- `float get_variable (int variable) const`

Return a specific variable for the particle system (see `VAR_*` enum).

- `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.

- `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.

- `void set_color_phase_pos (int phase, float pos)`

Set the position of a color phase (0 to 1).

- `float get_color_phase_pos (int phase) const`

Return the position of a color phase (0 to 1).

- `void set_color_phase_color (int phase, Color color)`

Set the color of a color phase.

- `Color get_color_phase_color (int phase) const`

Return the color of a color phase.

- `void set_material (Material material)`

Set the material used to draw particles.

- *Material* **get\_material()** const

Return the material used to draw particles.

- void **set\_emit\_timeout( float timeout )**
- *float* **get\_emit\_timeout()** const
- void **set\_height\_from\_velocity( bool enable )**
- *bool* **has\_height\_from\_velocity()** const
- void **set\_use\_local\_coordinates( bool enable )**
- *bool* **is\_using\_local\_coordinates()** const
- void **set\_color\_phases( int count )**
- *int* **get\_color\_phases()** const

## 9.187 Particles2D

**Inherits:** *Node2D* < *CanvasItem* < *Node* < *Object*

**Category:** Core

### 9.187.1 Brief Description

### 9.187.2 Member Functions

void	<i>set_emitting( bool active )</i>
<i>bool</i>	<i>is_emitting()</i> const
void	<i>set_amount( int amount )</i>
<i>int</i>	<i>get_amount()</i> const
void	<i>set_lifetime( float lifetime )</i>
<i>float</i>	<i>get_lifetime()</i> const
void	<i>set_time_scale( float time_scale )</i>
<i>float</i>	<i>get_time_scale()</i> const
void	<i>set_pre_process_time( float time )</i>
<i>float</i>	<i>get_pre_process_time()</i> const
void	<i>set_emit_timeout( float value )</i>
<i>float</i>	<i>get_emit_timeout()</i> const
void	<i>set_param( int param, float value )</i>
<i>float</i>	<i>get_param( int param )</i> const
void	<i>set_randomness( int param, float value )</i>
<i>float</i>	<i>get_randomness( int param )</i> const
<i>Texture</i>	<i>set_texture( Object texture )</i>
<i>Texture</i>	<i>get_texture()</i> const
void	<i>set_color( Color color )</i>
<i>Color</i>	<i>get_color()</i> const
<i>ColorRamp</i>	<i>set_color_ramp( Object color_ramp )</i>
<i>ColorRamp</i>	<i>get_color_ramp()</i> const
void	<i>set_emissor_offset( Vector2 offset )</i>

Continued on next page

Table 9.19 – continued from previous page

<i>Vector2</i>	<i>get_emissor_offset () const</i>
<i>void</i>	<i>set_flip_h (bool enable)</i>
<i>bool</i>	<i>is_flipped_h () const</i>
<i>void</i>	<i>set_flip_v (bool enable)</i>
<i>bool</i>	<i>is_flipped_v () const</i>
<i>void</i>	<i>set_h_frames (int enable)</i>
<i>int</i>	<i>get_h_frames () const</i>
<i>void</i>	<i>set_v_frames (int enable)</i>
<i>int</i>	<i>get_v_frames () const</i>
<i>void</i>	<i>set_emission_half_extents (Vector2 extents)</i>
<i>Vector2</i>	<i>get_emission_half_extents () const</i>
<i>void</i>	<i>set_color_phases (int phases)</i>
<i>int</i>	<i>get_color_phases () const</i>
<i>void</i>	<i>set_color_phase_color (int phase, Color color)</i>
<i>Color</i>	<i>get_color_phase_color (int phase) const</i>
<i>void</i>	<i>set_color_phase_pos (int phase, float pos)</i>
<i>float</i>	<i>get_color_phase_pos (int phase) const</i>
<i>void</i>	<i>pre_process (float time)</i>
<i>void</i>	<i>reset ()</i>
<i>void</i>	<i>set_use_local_space (bool enable)</i>
<i>bool</i>	<i>is_using_local_space () const</i>
<i>void</i>	<i>set_initial_velocity (Vector2 velocity)</i>
<i>Vector2</i>	<i>get_initial_velocity () const</i>
<i>void</i>	<i>set_explosiveness (float amount)</i>
<i>float</i>	<i>get_explosiveness () const</i>
<i>void</i>	<i>set_emission_points (Vector2Array points)</i>
<i>Vector2Array</i>	<i>get_emission_points () const</i>

### 9.187.3 Numeric Constants

- **PARAM\_DIRECTION = 0**
- **PARAM\_SPREAD = 1**
- **PARAM\_LINEAR\_VELOCITY = 2**
- **PARAM\_SPIN\_VELOCITY = 3**
- **PARAM\_ORBIT\_VELOCITY = 4**
- **PARAM\_GRAVITY\_DIRECTION = 5**
- **PARAM\_GRAVITY\_STRENGTH = 6**
- **PARAM\_RADIAL\_ACCEL = 7**
- **PARAM\_TANGENTIAL\_ACCEL = 8**
- **PARAM\_DAMPING = 9**
- **PARAM\_INITIAL\_ANGLE = 10**
- **PARAM\_INITIAL\_SIZE = 11**
- **PARAM\_FINAL\_SIZE = 12**
- **PARAM\_HUE\_VARIATION = 13**

- **PARAM\_ANIM\_SPEED\_SCALE** = 14
- **PARAM\_ANIM\_INITIAL\_POS** = 15
- **PARAM\_MAX** = 16
- **MAX\_COLOR\_PHASES** = 4

## 9.187.4 Member Function Description

- void **set\_emitting** ( *bool* active )
- *bool* **is\_emitting** ( ) const
- void **set\_amount** ( *int* amount )
- *int* **get\_amount** ( ) const
- void **set\_lifetime** ( *float* lifetime )
- *float* **get\_lifetime** ( ) const
- void **set\_time\_scale** ( *float* time\_scale )
- *float* **get\_time\_scale** ( ) const
- void **set\_pre\_process\_time** ( *float* time )
- *float* **get\_pre\_process\_time** ( ) const
- void **set\_emit\_timeout** ( *float* value )
- *float* **get\_emit\_timeout** ( ) const
- void **set\_param** ( *int* param, *float* value )
- *float* **get\_param** ( *int* param ) const
- void **set\_randomness** ( *int* param, *float* value )
- *float* **get\_randomness** ( *int* param ) const
- *Texture* **set\_texture** ( *Object* texture )
- *Texture* **get\_texture** ( ) const
- void **set\_color** ( *Color* color )
- *Color* **get\_color** ( ) const
- *ColorRamp* **set\_color\_ramp** ( *Object* color\_ramp )
- *ColorRamp* **get\_color\_ramp** ( ) const
- void **set\_emissor\_offset** ( *Vector2* offset )
- *Vector2* **get\_emissor\_offset** ( ) const
- void **set\_flip\_h** ( *bool* enable )
- *bool* **is\_flipped\_h** ( ) const
- void **set\_flip\_v** ( *bool* enable )
- *bool* **is\_flipped\_v** ( ) const
- void **set\_h\_frames** ( *int* enable )
- *int* **get\_h\_frames** ( ) const

- void **set\_v\_frames** ( *int* enable )
- *int* **get\_v\_frames** ( ) const
- void **set\_emission\_half\_extents** ( *Vector2* extents )
- *Vector2* **get\_emission\_half\_extents** ( ) const
- void **set\_color\_phases** ( *int* phases )
- *int* **get\_color\_phases** ( ) const
- void **set\_color\_phase\_color** ( *int* phase, *Color* color )
- *Color* **get\_color\_phase\_color** ( *int* phase ) const
- void **set\_color\_phase\_pos** ( *int* phase, *float* pos )
- *float* **get\_color\_phase\_pos** ( *int* phase ) const
- void **pre\_process** ( *float* time )
- void **reset** ( )
- void **set\_use\_local\_space** ( *bool* enable )
- *bool* **is\_using\_local\_space** ( ) const
- void **set\_initial\_velocity** ( *Vector2* velocity )
- *Vector2* **get\_initial\_velocity** ( ) const
- void **set\_explorativeness** ( *float* amount )
- *float* **get\_explorativeness** ( ) const
- void **set\_emission\_points** ( *Vector2Array* points )
- *Vector2Array* **get\_emission\_points** ( ) const

## 9.188 Patch9Frame

**Inherits:** *Control* < *CanvasItem* < *Node* < *Object*

**Category:** Core

### 9.188.1 Brief Description

### 9.188.2 Member Functions

void	<b>set_texture</b> ( <i>Object</i> texture )
<i>Object</i>	<b>get_texture</b> ( ) const
void	<b>set_modulate</b> ( <i>Color</i> modulate )
<i>Color</i>	<b>get_modulate</b> ( ) const
void	<b>set_patch_margin</b> ( <i>int</i> margin, <i>int</i> value )
<i>int</i>	<b>get_patch_margin</b> ( <i>int</i> margin ) const
void	<b>set_draw_center</b> ( <i>bool</i> draw_center )
<i>bool</i>	<b>get_draw_center</b> ( ) const

### 9.188.3 Member Function Description

- void `set_texture` ( *Object* texture )
- *Object* `get_texture` ( ) const
- void `set_modulate` ( *Color* modulate )
- *Color* `get_modulate` ( ) const
- void `set_patch_margin` ( *int* margin, *int* value )
- *int* `get_patch_margin` ( *int* margin ) const
- void `set_draw_center` ( *bool* draw\_center )
- *bool* `get_draw_center` ( ) const

## 9.189 Path

Inherits: *Spatial* < *Node* < *Object*

Category: Core

### 9.189.1 Brief Description

Container for a *Curve3D*.

### 9.189.2 Member Functions

void	<code>set_curve</code> ( <i>Curve3D</i> curve )
<i>Curve3D</i>	<code>get_curve</code> ( ) const

### 9.189.3 Description

This class is a container/Node-ification of a *Curve3D*, so it can have *Spatial* properties and *Node* info.

### 9.189.4 Member Function Description

- void `set_curve` ( *Curve3D* curve )

Sets the *Curve3D*.

- *Curve3D* `get_curve` ( ) const

Returns the *Curve3D* contained.

## 9.190 Path2D

Inherits: *Node2D* < *CanvasItem* < *Node* < *Object*

Category: Core

### 9.190.1 Brief Description

Container for a *Curve2D*.

### 9.190.2 Member Functions

<code>void</code>	<code>set_curve ( Curve2D curve )</code>
<code>Curve2D</code>	<code>get_curve ( ) const</code>

### 9.190.3 Description

This class is a container/Node-ification of a *Curve2D*, so it can have *Node2D* properties and *Node* info.

### 9.190.4 Member Function Description

- `void set_curve ( Curve2D curve )`

Sets the *Curve2D*.

- `Curve2D get_curve ( ) const`

Returns the *Curve2D* contained.

## 9.191 PathFollow

**Inherits:** *Spatial* < *Node* < *Object*

**Category:** Core

### 9.191.1 Brief Description

Point sampler for a *Path*.

### 9.191.2 Member Functions

<code>void</code>	<code>set_offset ( float offset )</code>
<code>float</code>	<code>get_offset ( ) const</code>
<code>void</code>	<code>set_h_offset ( float h_offset )</code>
<code>float</code>	<code>get_h_offset ( ) const</code>
<code>void</code>	<code>set_v_offset ( float v_offset )</code>
<code>float</code>	<code>get_v_offset ( ) const</code>
<code>void</code>	<code>set_unit_offset ( float unit_offset )</code>
<code>float</code>	<code>get_unit_offset ( ) const</code>
<code>void</code>	<code>set_rotation_mode ( int rotation_mode )</code>
<code>int</code>	<code>get_rotation_mode ( ) const</code>
<code>void</code>	<code>set_cubic_interpolation ( bool enable )</code>
<code>bool</code>	<code>get_cubic_interpolation ( ) const</code>
<code>void</code>	<code>set_loop ( bool loop )</code>
<code>bool</code>	<code>has_loop ( ) const</code>

### 9.191.3 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.

### 9.191.4 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.

### 9.191.5 Member Function Description

- `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.

- `float get_offset () const`

Returns the distance along the path in 3D units.

- `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.

- `float get_h_offset () const`

Returns the X displacement this node has from its parent *Path*.

- `void set_v_offset (float v_offset)`

Moves this node in the Y axis, for the same reasons of `set_h_offset`.

- `float get_v_offset () const`

Returns the Y displacement this node has from its parent *Path*.

- `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.

- `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).

- `void set_rotation_mode (int rotation_mode)`

Allows or forbids rotation on one or more axes, per the constants below.

- `int get_rotation_mode () const`

Returns the rotation mode. The constants below list which axes are allowed to rotate for each mode.

- 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 cubically.

- `bool get_cubic_interpolation () const`

This method returns whether the position between two cached points (see *set\_cubic\_interpolation*) is interpolated linearly, or cubically.

- `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.

- `bool has_loop () const`

Returns whether this node wraps its offsets around, or truncates them to the path ends.

## 9.192 PathFollow2D

**Inherits:** *Node2D* < *CanvasItem* < *Node* < *Object*

**Category:** Core

### 9.192.1 Brief Description

Point sampler for a *Path2D*.

### 9.192.2 Member Functions

void	<code>set_offset ( float offset )</code>
float	<code>get_offset () const</code>
void	<code>set_h_offset ( float h_offset )</code>
float	<code>get_h_offset () const</code>
void	<code>set_v_offset ( float v_offset )</code>
float	<code>get_v_offset () const</code>
void	<code>set_unit_offset ( float unit_offset )</code>
float	<code>get_unit_offset () const</code>
void	<code>set_rotate ( bool enable )</code>
bool	<code>is_rotating () const</code>
void	<code>set_cubic_interpolation ( bool enable )</code>
bool	<code>get_cubic_interpolation () const</code>
void	<code>set_loop ( bool loop )</code>
bool	<code>has_loop () const</code>

### 9.192.3 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.

### 9.192.4 Member Function Description

- `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.

- `float get_offset () const`

Returns the distance along the path in pixels.

- `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.

- `float get_h_offset () const`

Returns the horizontal displacement this node has from its parent [Path2D](#).

- `void set_v_offset (float v_offset)`

Moves the PathFollow2D vertically, for the same reasons of `set_h_offset`.

- `float get_v_offset () const`

Returns the vertical displacement this node has from its parent [Path2D](#).

- `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.

- `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).

- `void set_rotate (bool enable)`

If set, this node rotates to follow the path, making its descendants rotate.

- `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 cubically.

- `bool get_cubic_interpolation() const`

This method returns whether the position between two cached points (see `set_cubic_interpolation`) is interpolated linearly, or cubically.

- `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.

- `bool has_loop() const`

Returns whether this node wraps its offsets around, or truncates them to the path ends.

## 9.193 PathRemap

**Inherits:** `Object`

**Category:** Core

### 9.193.1 Brief Description

Singleton containing the list of remapped resources.

### 9.193.2 Member Functions

<code>void</code>	<code>add_remap( String from, String to, String locale="" )</code>
<code>bool</code>	<code>has_remap( String path ) const</code>
<code>String</code>	<code>get_remap( String path ) const</code>
<code>void</code>	<code>erase_remap( String path )</code>
<code>void</code>	<code>clear_remaps()</code>

### 9.193.3 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.

### 9.193.4 Member Function Description

- `void add_remap( String from, String to, String locale="" )`

Add a remap from a file to another.

- `bool has_remap( String path ) const`

Return true if a file is being remapped.

- `String get_remap( String path ) const`

Return the remapped new path of a file.

- `void erase_remap( String path )`

Erase a remap.

- void **clear\_remaps()**

Clear all remaps.

## 9.194 PCKPacker

**Inherits:** *Reference < Object*

**Category:** Core

### 9.194.1 Brief Description

### 9.194.2 Member Functions

<i>int</i>	<b>pck_start</b> ( <i>String</i> pck_name, <i>int</i> alignment )
<i>int</i>	<b>add_file</b> ( <i>String</i> pck_path, <i>String</i> source_path )
<i>int</i>	<b>flush</b> ( <i>bool</i> verbose )

### 9.194.3 Member Function Description

- *int* **pck\_start** ( *String* pck\_name, *int* alignment )
- *int* **add\_file** ( *String* pck\_path, *String* source\_path )
- *int* **flush** ( *bool* verbose )

## 9.195 Performance

**Inherits:** *Object*

**Category:** Core

### 9.195.1 Brief Description

### 9.195.2 Member Functions

<i>float</i>	<b>get_monitor</b> ( <i>int</i> monitor ) const
--------------	---

### 9.195.3 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**

#### 9.195.4 Member Function Description

- `float get_monitor ( int monitor ) const`

### 9.196 PHashTranslation

Inherits: *Translation < Resource < Reference < Object*

Category: Core

#### 9.196.1 Brief Description

Optimized translation.

#### 9.196.2 Member Functions

void	<code>generate ( Translation from )</code>
------	--

### 9.196.3 Description

Optimized translation. Uses real-time compressed translations, which results in very small dictionaries.

### 9.196.4 Member Function Description

- void **generate** ( *Translation* from )

## 9.197 Physics2DDirectBodyState

**Inherits:** *Object*

**Inherited By:** *Physics2DDirectBodyStateSW*

**Category:** Core

### 9.197.1 Brief Description

Direct access object to a physics body in the *Physics2DServer*.

### 9.197.2 Member Functions

<i>Vector2</i>	<code>get_total_gravity () const</code>
<i>float</i>	<code>get_total_linear_damp () const</code>
<i>float</i>	<code>get_total_angular_damp () const</code>
<i>float</i>	<code>get_inverse_mass () const</code>
<i>float</i>	<code>get_inverse_inertia () const</code>
<i>void</i>	<code>set_linear_velocity ( <i>Vector2</i> velocity )</code>
<i>Vector2</i>	<code>get_linear_velocity () const</code>
<i>void</i>	<code>set_angular_velocity ( <i>float</i> velocity )</code>
<i>float</i>	<code>get_angular_velocity () const</code>
<i>void</i>	<code>set_transform ( <i>Matrix32</i> transform )</code>
<i>Matrix32</i>	<code>get_transform () const</code>
<i>void</i>	<code>set_sleep_state ( <i>bool</i> enabled )</code>
<i>bool</i>	<code>is_sleeping () const</code>
<i>int</i>	<code>get_contact_count () const</code>
<i>Vector2</i>	<code>get_contact_local_pos ( <i>int</i> contact_idx ) const</code>
<i>Vector2</i>	<code>get_contact_local_normal ( <i>int</i> contact_idx ) const</code>
<i>int</i>	<code>get_contact_local_shape ( <i>int</i> contact_idx ) const</code>
<i>RID</i>	<code>get_contact.collider ( <i>int</i> contact_idx ) const</code>
<i>Vector2</i>	<code>get_contact.collider_pos ( <i>int</i> contact_idx ) const</code>
<i>int</i>	<code>get_contact.collider_id ( <i>int</i> contact_idx ) const</code>
<i>Object</i>	<code>get_contact.collider_object ( <i>int</i> contact_idx ) const</code>
<i>int</i>	<code>get_contact.collider_shape ( <i>int</i> contact_idx ) const</code>
<i>Variant</i>	<code>get_contact.collider_shape_metadata ( <i>int</i> contact_idx ) const</code>
<i>Vector2</i>	<code>get_contact.collider_velocity_at_pos ( <i>int</i> contact_idx ) const</code>
<i>float</i>	<code>get_step () const</code>
<i>void</i>	<code>integrate_forces ()</code>
<i>Physics2DDirectSpaceState</i>	<code>get_space_state ()</code>

### 9.197.3 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.

### 9.197.4 Member Function Description

- `Vector2 get_total_gravity () const`

Return the total gravity vector being currently applied to this body.

- `float get_total_linear_damp () const`
- `float get_total_angular_damp () const`
- `float get_inverse_mass () const`

Return the inverse of the mass of the body.

- `float get_inverse_inertia () const`

Return the inverse of the inertia of the body.

- `void set_linear_velocity ( Vector2 velocity )`

Change the linear velocity of the body.

- `Vector2 get_linear_velocity () const`

Return the current linear velocity of the body.

- `void set_angular_velocity ( float velocity )`

Change the angular velocity of the body.

- `float get_angular_velocity () const`

Return the angular velocity of the body.

- `void set_transform ( Matrix32 transform )`

Change the transform matrix of the body.

- `Matrix32 get_transform () const`

Return the transform matrix of the body.

- `void set_sleep_state ( bool enabled )`

Set the sleeping state of the body, only affects character/rigid bodies.

- `bool is_sleeping () const`

Return true if this body is currently sleeping (not active).

- `int get_contact_count () 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_contact_local_pos ( int contact_idx ) const`

Return the local position (of this body) of the contact point.

- `Vector2 get_contact_local_normal ( int contact_idx ) const`

- `int get_contact_local_shape ( int contact_idx ) const`

Return the local shape index of the collision.

- `RID get_contact.collider ( int contact_idx ) const`

Return the RID of the collider.

- `Vector2 get_contact.collider_pos ( int contact_idx ) const`

Return the contact position in the collider.

- `int get_contact.collider_id ( int contact_idx ) const`

Return the object id of the collider.

- `Object get_contact.collider_object ( 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).

- `int get_contact.collider_shape ( int contact_idx ) const`

Return the collider shape index.

- Variant `get_contact.collider_shape_metadata ( int contact_idx ) const`

- `Vector2 get_contact.collider_velocity_at_pos ( int contact_idx ) const`

Return the linear velocity vector at contact point of the collider.

- `float get_step ( ) const`

Return the timestep (delta) used for the simulation.

- `void integrate_forces ( )`

Call the built-in force integration code.

- `Physics2DDirectSpaceState get_space_state ( )`

Return the current state of space, useful for queries.

## 9.198 Physics2DDirectBodyStateSW

**Inherits:** `Physics2DDirectBodyState < Object`

**Category:** Core

### 9.198.1 Brief Description

## 9.199 Physics2DDirectSpaceState

**Inherits:** `Object`

**Category:** Core

### 9.199.1 Brief Description

Direct access object to a space in the `Physics2DServer`.

## 9.199.2 Member Functions

<code>Array</code>	<code>intersect_point ( Vector2 point, int max_results=32, Array exclude=Array(), int layer_mask=2147483647, int type_mask=15 )</code>
<code>Dictionary</code>	<code>intersect_ray ( Vector2 from, Vector2 to, Array exclude=Array(), int layer_mask=2147483647, int type_mask=15 )</code>
<code>Array</code>	<code>intersect_shape ( Physics2DShapeQueryParameters shape, int max_results=32 )</code>
<code>Array</code>	<code>cast_motion ( Physics2DShapeQueryParameters shape )</code>
<code>Array</code>	<code>collide_shape ( Physics2DShapeQueryParameters shape, int max_results=32 )</code>
<code>Dictionary</code>	<code>get_rest_info ( Physics2DShapeQueryParameters shape )</code>

## 9.199.3 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`

## 9.199.4 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.

## 9.199.5 Member Function Description

- `Array intersect_point ( Vector2 point, int max_results=32, Array exclude=Array(), int layer_mask=2147483647, int type_mask=15 )`
- `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 of the object against which the ray was stopped.

collider\_: collider against which the ray was stopped.

collider\_id: collider id of the object against which the ray was stopped.

collider: 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.

- `Array intersect_shape ( Physics2DShapeQueryParameters shape, int max_results=32 )`

Intersect a given shape (RID or *Shape2D*) against the space, the intersected shapes are returned in a special result object.

- `Array cast_motion ( Physics2DShapeQueryParameters shape )`
- `Array collide_shape ( Physics2DShapeQueryParameters shape, int max_results=32 )`
- `Dictionary get_rest_info ( Physics2DShapeQueryParameters shape )`

## 9.200 Physics2DServer

**Inherits:** *Object*

**Inherited By:** *Physics2DServerSW*

**Category:** Core

### 9.200.1 Brief Description

Physics 2D Server.

### 9.200.2 Member Functions

<i>RID</i>	<code>shape_create ( int type )</code>
void	<code>shape_set_data ( RID shape, var data )</code>
<i>int</i>	<code>shape_get_type ( RID shape ) const</code>
void	<code>shape_get_data ( RID shape ) const</code>
<i>RID</i>	<code>space_create ()</code>
void	<code>space_set_active ( RID space, bool active )</code>
<i>bool</i>	<code>space_is_active ( RID space ) const</code>
void	<code>space_set_param ( RID space, int param, float value )</code>
<i>float</i>	<code>space_get_param ( RID space, int param ) const</code>
<i>Physics2DDirectSpaceState</i>	<code>space_get_direct_state ( RID space )</code>
<i>RID</i>	<code>area_create ()</code>
void	<code>area_set_space ( RID area, RID space )</code>
<i>RID</i>	<code>area_get_space ( RID area ) const</code>
void	<code>area_set_space_override_mode ( RID area, int mode )</code>
<i>int</i>	<code>area_get_space_override_mode ( RID area ) const</code>
void	<code>area_add_shape ( RID area, RID shape, Matrix32 transform=1,0,0,1,0,0 )</code>
void	<code>area_set_shape ( RID area, int shape_idx, RID shape )</code>
void	<code>area_set_shape_transform ( RID area, int shape_idx, Matrix32 transform )</code>
<i>int</i>	<code>area_get_shape_count ( RID area ) const</code>
<i>RID</i>	<code>area_get_shape ( RID area, int shape_idx ) const</code>
<i>Matrix32</i>	<code>area_get_shape_transform ( RID area, int shape_idx ) const</code>
void	<code>area_remove_shape ( RID area, int shape_idx )</code>
void	<code>area_clear_shapes ( RID area )</code>
void	<code>area_set_layer_mask ( RID area, int mask )</code>
void	<code>area_set_collision_mask ( RID area, int mask )</code>
void	<code>area_set_param ( RID area, int param, var value )</code>
void	<code>area_set_transform ( RID area, Matrix32 transform )</code>

Contin

Table 9.20 – continued from previous page

void	<code>area_get_param ( RID area, int param ) const</code>
<i>Matrix32</i>	<code>area_get_transform ( RID area ) const</code>
void	<code>area_attach_object_instance_ID ( RID area, int id )</code>
<i>int</i>	<code>area_get_object_instance_ID ( RID area ) const</code>
void	<code>area_set_monitor_callback ( RID area, Object receiver, String method )</code>
<i>RID</i>	<code>body_create ( int mode=2, bool init_sleeping=false )</code>
void	<code>body_set_space ( RID body, RID space )</code>
<i>RID</i>	<code>body_get_space ( RID body ) const</code>
void	<code>body_set_mode ( RID body, int mode )</code>
<i>int</i>	<code>body_get_mode ( RID body ) const</code>
void	<code>body_add_shape ( RID body, RID shape, Matrix32 transform=1,0,0,1,0,0 )</code>
void	<code>body_set_shape ( RID body, int shape_idx, RID shape )</code>
void	<code>body_set_shape_transform ( RID body, int shape_idx, Matrix32 transform )</code>
void	<code>body_set_shape_metadata ( RID body, int shape_idx, var metadata )</code>
<i>int</i>	<code>body_get_shape_count ( RID body ) const</code>
<i>RID</i>	<code>body_get_shape ( RID body, int shape_idx ) const</code>
<i>Matrix32</i>	<code>body_get_shape_transform ( RID body, int shape_idx ) const</code>
void	<code>body_get_shape_metadata ( RID body, int shape_idx ) const</code>
void	<code>body_remove_shape ( RID body, int shape_idx )</code>
void	<code>body_clear_shapes ( RID body )</code>
void	<code>body_set_shape_as_trigger ( RID body, int shape_idx, bool enable )</code>
<i>bool</i>	<code>body_is_shape_set_as_trigger ( RID body, int shape_idx ) const</code>
void	<code>body_attach_object_instance_ID ( RID body, int id )</code>
<i>int</i>	<code>body_get_object_instance_ID ( RID body ) const</code>
void	<code>body_set_continuous_collision_detection_mode ( RID body, int mode )</code>
<i>int</i>	<code>body_get_continuous_collision_detection_mode ( RID body ) const</code>
void	<code>body_set_layer_mask ( RID body, int mask )</code>
<i>int</i>	<code>body_get_layer_mask ( RID body ) const</code>
void	<code>body_set_collision_mask ( RID body, int mask )</code>
<i>int</i>	<code>body_get_collision_mask ( RID body ) const</code>
void	<code>body_set_param ( RID body, int param, float value )</code>
<i>float</i>	<code>body_get_param ( RID body, int param ) const</code>
void	<code>body_set_state ( RID body, int state, var value )</code>
void	<code>body_get_state ( RID body, int state ) const</code>
void	<code>body_apply_impulse ( RID body, Vector2 pos, Vector2 impulse )</code>
void	<code>body_set_axis_velocity ( RID body, Vector2 axis_velocity )</code>
void	<code>body_add_collision_exception ( RID body, RID excepted_body )</code>
void	<code>body_remove_collision_exception ( RID body, RID excepted_body )</code>
void	<code>body_set_max_contacts_reported ( RID body, int amount )</code>
<i>int</i>	<code>body_get_max_contacts_reported ( RID body ) const</code>
void	<code>body_set_one_way_collision_direction ( RID body, Vector2 normal )</code>
<i>Vector2</i>	<code>body_get_one_way_collision_direction ( RID body ) const</code>
void	<code>body_set_one_way_collision_max_depth ( RID body, float depth )</code>
<i>float</i>	<code>body_get_one_way_collision_max_depth ( RID body ) const</code>
void	<code>body_set OMIT force_integration ( RID body, bool enable )</code>
<i>bool</i>	<code>body_is omitting force_integration ( RID body ) const</code>
void	<code>body_set_force_integration_callback ( RID body, Object receiver, String method, var userdata=NULL )</code>
<i>bool</i>	<code>body_test_motion ( RID body, Vector2 motion, float margin=0.08, Physics2DTestMotionResult result=NULL )</code>
void	<code>joint_set_param ( RID joint, int param, float value )</code>
<i>float</i>	<code>joint_get_param ( RID joint, int param ) const</code>

Contin

Table 9.20 – continued from previous page

<i>RID</i>	<i>pin_joint_create</i> ( <i>Vector2</i> anchor, <i>RID</i> body_a, <i>RID</i> body_b=RID() )
<i>RID</i>	<i>groove_joint_create</i> ( <i>Vector2</i> groove1_a, <i>Vector2</i> groove2_a, <i>Vector2</i> anchor_b, <i>RID</i> body_a=RID(), <i>RID</i> body_b=RID() )
<i>RID</i>	<i>damped_spring_joint_create</i> ( <i>Vector2</i> anchor_a, <i>Vector2</i> anchor_b, <i>RID</i> body_a, <i>RID</i> body_b=RID() )
void	<i>damped_string_joint_set_param</i> ( <i>RID</i> joint, <i>int</i> param, <i>float</i> value=RID() )
<i>float</i>	<i>damped_string_joint_get_param</i> ( <i>RID</i> joint, <i>int</i> param ) const
<i>int</i>	<i>joint_get_type</i> ( <i>RID</i> joint ) const
void	<i>free_rid</i> ( <i>RID</i> rid )
void	<i>set_active</i> ( <i>bool</i> active )
<i>int</i>	<i>get_process_info</i> ( <i>int</i> process_info )

### 9.200.3 Numeric Constants

- **SHAPE\_LINE = 0**
- **SHAPE\_SEGMENT = 2**
- **SHAPE\_CIRCLE = 3**
- **SHAPE\_RECTANGLE = 4**
- **SHAPE\_CAPSULE = 5**
- **SHAPE\_CONVEX\_POLYGON = 6**
- **SHAPE\_CONCAVE\_POLYGON = 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\_LINEAR\_DAMP** = 4
- **BODY\_PARAM\_ANGULAR\_DAMP** = 5
- **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
- **JOINT\_PIN** = 0
- **JOINT\_GROOVE** = 1
- **JOINT DAMPED\_SPRING** = 2
- **DAMPED\_STRING\_REST\_LENGTH** = 0
- **DAMPED\_STRING\_STIFFNESS** = 1
- **DAMPED\_STRING\_DAMPING** = 2
- **CCD\_MODE\_DISABLED** = 0
- **CCD\_MODE\_CAST\_RAY** = 1
- **CCD\_MODE\_CAST\_SHAPE** = 2
- **AREA\_BODY\_ADDED** = 0
- **AREA\_BODY\_REMOVED** = 1
- **INFO\_ACTIVE\_OBJECTS** = 0
- **INFO\_COLLISION\_PAIRS** = 1
- **INFO\_ISLAND\_COUNT** = 2

#### 9.200.4 Description

Physics 2D Server is the server responsible for all 2D physics.

#### 9.200.5 Member Function Description

- *RID* **shape\_create** ( *int* type )
- void **shape\_set\_data** ( *RID* shape, var data )
- *int* **shape\_get\_type** ( *RID* shape ) const

- void **shape\_get\_data** ( *RID* shape ) const
- *RID* **space\_create** ( )
- void **space\_set\_active** ( *RID* space, *bool* active )
- *bool* **space\_is\_active** ( *RID* space ) const
- void **space\_set\_param** ( *RID* space, *int* param, *float* value )
- *float* **space\_get\_param** ( *RID* space, *int* param ) const
- *Physics2DDirectSpaceState* **space\_get\_direct\_state** ( *RID* space )
- *RID* **area\_create** ( )
- void **area\_set\_space** ( *RID* area, *RID* space )
- *RID* **area\_get\_space** ( *RID* area ) const
- void **area\_set\_space\_override\_mode** ( *RID* area, *int* mode )
- *int* **area\_get\_space\_override\_mode** ( *RID* area ) const
- void **area\_add\_shape** ( *RID* area, *RID* shape, *Matrix32* transform=1,0, 0,1, 0,0 )
- void **area\_set\_shape** ( *RID* area, *int* shape\_idx, *RID* shape )
- void **area\_set\_shape\_transform** ( *RID* area, *int* shape\_idx, *Matrix32* transform )
- *int* **area\_get\_shape\_count** ( *RID* area ) const
- *RID* **area\_get\_shape** ( *RID* area, *int* shape\_idx ) const
- *Matrix32* **area\_get\_shape\_transform** ( *RID* area, *int* shape\_idx ) const
- void **area\_remove\_shape** ( *RID* area, *int* shape\_idx )
- void **area\_clear\_shapes** ( *RID* area )
- void **area\_set\_layer\_mask** ( *RID* area, *int* mask )
- void **area\_set\_collision\_mask** ( *RID* area, *int* mask )
- void **area\_set\_param** ( *RID* area, *int* param, var value )
- void **area\_set\_transform** ( *RID* area, *Matrix32* transform )
- void **area\_get\_param** ( *RID* area, *int* param ) const
- *Matrix32* **area\_get\_transform** ( *RID* area ) const
- void **area\_attach\_object\_instance\_ID** ( *RID* area, *int* id )
- *int* **area\_get\_object\_instance\_ID** ( *RID* area ) const
- void **area\_set\_monitor\_callback** ( *RID* area, *Object* receiver, *String* method )
- *RID* **body\_create** ( *int* mode=2, *bool* init\_sleeping=false )
- void **body\_set\_space** ( *RID* body, *RID* space )
- *RID* **body\_get\_space** ( *RID* body ) const
- void **body\_set\_mode** ( *RID* body, *int* mode )
- *int* **body\_get\_mode** ( *RID* body ) const
- void **body\_add\_shape** ( *RID* body, *RID* shape, *Matrix32* transform=1,0, 0,1, 0,0 )
- void **body\_set\_shape** ( *RID* body, *int* shape\_idx, *RID* shape )

- void **body\_set\_shape\_transform** ( *RID* body, *int* shape\_idx, *Matrix32* transform )
- void **body\_set\_shape\_metadata** ( *RID* body, *int* shape\_idx, var metadata )
- *int* **body\_get\_shape\_count** ( *RID* body ) const
- *RID* **body\_get\_shape** ( *RID* body, *int* shape\_idx ) const
- *Matrix32* **body\_get\_shape\_transform** ( *RID* body, *int* shape\_idx ) const
- void **body\_get\_shape\_metadata** ( *RID* body, *int* shape\_idx ) const
- void **body\_remove\_shape** ( *RID* body, *int* shape\_idx )
- void **body\_clear\_shapes** ( *RID* body )
- void **body\_set\_shape\_as\_trigger** ( *RID* body, *int* shape\_idx, *bool* enable )
- *bool* **body\_is\_shape\_set\_as\_trigger** ( *RID* body, *int* shape\_idx ) const
- void **body\_attach\_object\_instance\_ID** ( *RID* body, *int* id )
- *int* **body\_get\_object\_instance\_ID** ( *RID* body ) const
- void **body\_set\_continuous\_collision\_detection\_mode** ( *RID* body, *int* mode )
- *int* **body\_get\_continuous\_collision\_detection\_mode** ( *RID* body ) const
- void **body\_set\_layer\_mask** ( *RID* body, *int* mask )
- *int* **body\_get\_layer\_mask** ( *RID* body ) const
- void **body\_set\_collision\_mask** ( *RID* body, *int* mask )
- *int* **body\_get\_collision\_mask** ( *RID* body ) const
- void **body\_set\_param** ( *RID* body, *int* param, *float* value )
- *float* **body\_get\_param** ( *RID* body, *int* param ) const
- void **body\_set\_state** ( *RID* body, *int* state, var value )
- void **body\_get\_state** ( *RID* body, *int* state ) const
- void **body\_apply\_impulse** ( *RID* body, *Vector2* pos, *Vector2* impulse )
- void **body\_set\_axis\_velocity** ( *RID* body, *Vector2* axis\_velocity )
- void **body\_add\_collision\_exception** ( *RID* body, *RID* excepted\_body )
- void **body\_remove\_collision\_exception** ( *RID* body, *RID* excepted\_body )
- void **body\_set\_max\_contacts\_reported** ( *RID* body, *int* amount )
- *int* **body\_get\_max\_contacts\_reported** ( *RID* body ) const
- void **body\_set\_one\_way\_collision\_direction** ( *RID* body, *Vector2* normal )
- *Vector2* **body\_get\_one\_way\_collision\_direction** ( *RID* body ) const
- void **body\_set\_one\_way\_collision\_max\_depth** ( *RID* body, *float* depth )
- *float* **body\_get\_one\_way\_collision\_max\_depth** ( *RID* body ) const
- void **body\_set OMIT force\_integration** ( *RID* body, *bool* enable )
- *bool* **body\_is omitting force\_integration** ( *RID* body ) const
- void **body\_set\_force\_integration\_callback** ( *RID* body, *Object* receiver, *String* method, var userdata=NULL )

- `bool body_test_motion ( RID body, Vector2 motion, float margin=0.08, Physics2DTestMotionResult result=NULL )`
- `void joint_set_param ( RID joint, int param, float value )`
- `float joint_get_param ( RID joint, int param ) const`
- `RID pin_joint_create ( Vector2 anchor, RID body_a, RID body_b=RID() )`
- `RID groove_joint_create ( Vector2 groove1_a, Vector2 groove2_a, Vector2 anchor_b, RID body_a=RID(), RID body_b=RID() )`
- `RID damped_spring_joint_create ( Vector2 anchor_a, Vector2 anchor_b, RID body_a, RID body_b=RID() )`
- `void damped_string_joint_set_param ( RID joint, int param, float value=RID() )`
- `float damped_string_joint_get_param ( RID joint, int param ) const`
- `int joint_get_type ( RID joint ) const`
- `void free_rid ( RID rid )`
- `void set_active ( bool active )`
- `int get_process_info ( int process_info )`

## 9.201 Physics2DServerSW

**Inherits:** *Physics2DServer < Object*

**Category:** Core

### 9.201.1 Brief Description

## 9.202 Physics2DShapeQueryParameters

**Inherits:** *Reference < Object*

**Category:** Core

## 9.202.1 Brief Description

## 9.202.2 Member Functions

void	<code>set_shape ( Shape2D shape )</code>
void	<code>set_shape_rid ( RID shape )</code>
<i>RID</i>	<code>get_shape_rid () const</code>
void	<code>set_transform ( Matrix32 transform )</code>
<i>Matrix32</i>	<code>get_transform () const</code>
void	<code>set_motion ( Vector2 motion )</code>
<i>Vector2</i>	<code>get_motion () const</code>
void	<code>set_margin ( float margin )</code>
<i>float</i>	<code>get_margin () const</code>
void	<code>set_layer_mask ( int layer_mask )</code>
<i>int</i>	<code>get_layer_mask () const</code>
void	<code>set_object_type_mask ( int object_type_mask )</code>
<i>int</i>	<code>get_object_type_mask () const</code>
void	<code>set_exclude ( Array exclude )</code>
<i>Array</i>	<code>get_exclude () const</code>

## 9.202.3 Member Function Description

- void `set_shape ( Shape2D shape )`
- void `set_shape_rid ( RID shape )`
- *RID* `get_shape_rid () const`
- void `set_transform ( Matrix32 transform )`
- *Matrix32* `get_transform () const`
- void `set_motion ( Vector2 motion )`
- *Vector2* `get_motion () const`
- void `set_margin ( float margin )`
- *float* `get_margin () const`
- void `set_layer_mask ( int layer_mask )`
- *int* `get_layer_mask () const`
- void `set_object_type_mask ( int object_type_mask )`
- *int* `get_object_type_mask () const`
- void `set_exclude ( Array exclude )`
- *Array* `get_exclude () const`

## 9.203 Physics2DShapeQueryResult

**Inherits:** [Reference](#) < [Object](#)

**Category:** Core

### 9.203.1 Brief Description

### 9.203.2 Member Functions

<i>int</i>	<code>get_result_count () const</code>
<i>RID</i>	<code>get_result_rid ( int idx ) const</code>
<i>int</i>	<code>get_result_object_id ( int idx ) const</code>
<i>Object</i>	<code>get_result_object ( int idx ) const</code>
<i>int</i>	<code>get_result_object_shape ( int idx ) const</code>

### 9.203.3 Member Function Description

- `int get_result_count () const`
- `RID get_result_rid ( int idx ) const`
- `int get_result_object_id ( int idx ) const`
- `Object get_result_object ( int idx ) const`
- `int get_result_object_shape ( int idx ) const`

## 9.204 Physics2DTestMotionResult

Inherits: [Reference](#) < [Object](#)

Category: Core

### 9.204.1 Brief Description

### 9.204.2 Member Functions

<i>Vector2</i>	<code>get_motion () const</code>
<i>Vector2</i>	<code>get_motion_remainder () const</code>
<i>Vector2</i>	<code>get_collision_point () const</code>
<i>Vector2</i>	<code>get_collision_normal () const</code>
<i>Vector2</i>	<code>get.collider_velocity () const</code>
<i>int</i>	<code>get.collider_id () const</code>
<i>RID</i>	<code>get.collider_rid () const</code>
<i>Object</i>	<code>get.collider () const</code>
<i>int</i>	<code>get.collider_shape () const</code>

### 9.204.3 Member Function Description

- `Vector2 get_motion () const`
- `Vector2 get_motion_remainder () const`
- `Vector2 get_collision_point () const`
- `Vector2 get_collision_normal () const`
- `Vector2 get.collider_velocity () const`

- `int get_collider_id () const`
- `RID get_collider_rid () const`
- `Object get_collider () const`
- `int get_collider_shape () const`

## 9.205 PhysicsBody

**Inherits:** `CollisionObject < Spatial < Node < Object`

**Inherited By:** `VehicleBody, KinematicBody, StaticBody, RigidBody`

**Category:** Core

### 9.205.1 Brief Description

Base class for different types of Physics bodies.

### 9.205.2 Member Functions

<code>void</code>	<code>set_layer_mask ( int mask )</code>
<code>int</code>	<code>get_layer_mask () const</code>
<code>void</code>	<code>add_collision_exception_with ( PhysicsBody body )</code>
<code>void</code>	<code>remove_collision_exception_with ( PhysicsBody body )</code>

### 9.205.3 Description

PhysicsBody is an abstract base class for implementing a physics body. All PhysicsBody types inherit from it.

### 9.205.4 Member Function Description

- `void set_layer_mask ( int mask )`
- `int get_layer_mask () const`
- `void add_collision_exception_with ( PhysicsBody body )`
- `void remove_collision_exception_with ( PhysicsBody body )`

## 9.206 PhysicsBody2D

**Inherits:** `CollisionObject2D < Node2D < CanvasItem < Node < Object`

**Inherited By:** `RigidBody2D, StaticBody2D, KinematicBody2D`

**Category:** Core

### 9.206.1 Brief Description

Base class for all objects affected by physics.

### 9.206.2 Member Functions

void	<code>set_layer_mask ( int mask )</code>
<i>int</i>	<code>get_layer_mask () const</code>
void	<code>set_collision_mask ( int mask )</code>
<i>int</i>	<code>get_collision_mask () const</code>
void	<code>set_collision_mask_bit ( int bit, bool value )</code>
<i>bool</i>	<code>get_collision_mask_bit ( int bit ) const</code>
void	<code>set_layer_mask_bit ( int bit, bool value )</code>
<i>bool</i>	<code>get_layer_mask_bit ( int bit ) const</code>
void	<code>set_one_way_collision_direction ( Vector2 dir )</code>
<i>Vector2</i>	<code>get_one_way_collision_direction () const</code>
void	<code>set_one_way_collision_max_depth ( float depth )</code>
<i>float</i>	<code>get_one_way_collision_max_depth () const</code>
void	<code>add_collision_exception_with ( PhysicsBody2D body )</code>
void	<code>remove_collision_exception_with ( PhysicsBody2D body )</code>

### 9.206.3 Description

PhysicsBody2D is an abstract base class for implementing a physics body. All \*Body2D types inherit from it.

### 9.206.4 Member Function Description

- 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.

- *int* `get_layer_mask () const`

Return the physics layer this area is in.

- void `set_collision_mask ( int mask )`

Set the physics layers this area can scan for collisions.

- *int* `get_collision_mask () const`

Return 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.

- *bool* `get_collision_mask_bit ( int bit ) const`

Return an individual bit on the collision mask.

- 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.

- `bool get_layer_mask_bit ( int bit ) const`

Return an individual bit on the collision mask.

- `void set_one_way_collision_direction ( Vector2 dir )`

Set a direction in which bodies can go through this one. If this value is different from (0,0), any movement within 90 degrees of this vector is considered a valid movement. Set this direction to (0,0) to disable one-way collisions.

- `Vector2 get_one_way_collision_direction ( ) const`

Return the direction used for one-way collision detection.

- `void set_one_way_collision_max_depth ( float depth )`

Set how far a body can go through this one, when it allows one-way collisions (see `set_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 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.

- `void remove_collision_exception_with ( PhysicsBody2D body )`

Removes a body from the collision exception list.

## 9.207 PhysicsDirectBodyState

**Inherits:** `Object`

**Inherited By:** `PhysicsDirectBodyStateSW`

**Category:** Core

## 9.207.1 Brief Description

## 9.207.2 Member Functions

<i>Vector3</i>	<code>get_total_gravity () const</code>
<i>float</i>	<code>get_total_linear_damp () const</code>
<i>float</i>	<code>get_total_angular_damp () const</code>
<i>float</i>	<code>get_inverse_mass () const</code>
<i>Vector3</i>	<code>get_inverse_inertia () const</code>
<i>void</i>	<code>set_linear_velocity ( <i>Vector3</i> velocity )</code>
<i>Vector3</i>	<code>get_linear_velocity () const</code>
<i>void</i>	<code>set_angular_velocity ( <i>Vector3</i> velocity )</code>
<i>Vector3</i>	<code>get_angular_velocity () const</code>
<i>void</i>	<code>set_transform ( <i>Transform</i> transform )</code>
<i>Transform</i>	<code>get_transform () const</code>
<i>void</i>	<code>add_force ( <i>Vector3</i> force, <i>Vector3</i> pos )</code>
<i>void</i>	<code>apply_impulse ( <i>Vector3</i> pos, <i>Vector3</i> j )</code>
<i>void</i>	<code>set_sleep_state ( <i>bool</i> enabled )</code>
<i>bool</i>	<code>is_sleeping () const</code>
<i>int</i>	<code>get_contact_count () const</code>
<i>Vector3</i>	<code>get_contact_local_pos ( <i>int</i> contact_idx ) const</code>
<i>Vector3</i>	<code>get_contact_local_normal ( <i>int</i> contact_idx ) const</code>
<i>int</i>	<code>get_contact_local_shape ( <i>int</i> contact_idx ) const</code>
<i>RID</i>	<code>get_contact.collider ( <i>int</i> contact_idx ) const</code>
<i>Vector3</i>	<code>get_contact.collider_pos ( <i>int</i> contact_idx ) const</code>
<i>int</i>	<code>get_contact.collider_id ( <i>int</i> contact_idx ) const</code>
<i>Object</i>	<code>get_contact.collider_object ( <i>int</i> contact_idx ) const</code>
<i>int</i>	<code>get_contact.collider_shape ( <i>int</i> contact_idx ) const</code>
<i>Vector3</i>	<code>get_contact.collider_velocity_at_pos ( <i>int</i> contact_idx ) const</code>
<i>float</i>	<code>get_step () const</code>
<i>void</i>	<code>integrate_forces ()</code>
<i>PhysicsDirectSpaceState</i>	<code>get_space_state ()</code>

## 9.207.3 Member Function Description

- `Vector3 get_total_gravity () const`
- `float get_total_linear_damp () const`
- `float get_total_angular_damp () const`
- `float get_inverse_mass () const`
- `Vector3 get_inverse_inertia () const`
- `void set_linear_velocity ( Vector3 velocity )`
- `Vector3 get_linear_velocity () const`
- `void set_angular_velocity ( Vector3 velocity )`
- `Vector3 get_angular_velocity () const`
- `void set_transform ( Transform transform )`
- `Transform get_transform () const`

- void **add\_force** ( *Vector3* force, *Vector3* pos )
- void **apply\_impulse** ( *Vector3* pos, *Vector3* j )
- void **set\_sleep\_state** ( *bool* enabled )
- *bool* **is\_sleeping** ( ) const
- *int* **get\_contact\_count** ( ) const
- *Vector3* **get\_contact\_local\_pos** ( *int* contact\_idx ) const
- *Vector3* **get\_contact\_local\_normal** ( *int* contact\_idx ) const
- *int* **get\_contact\_local\_shape** ( *int* contact\_idx ) const
- *RID* **get\_contact.collider** ( *int* contact\_idx ) const
- *Vector3* **get\_contact.collider\_pos** ( *int* contact\_idx ) const
- *int* **get\_contact.collider\_id** ( *int* contact\_idx ) const
- *Object* **get\_contact.collider\_object** ( *int* contact\_idx ) const
- *int* **get\_contact.collider\_shape** ( *int* contact\_idx ) const
- *Vector3* **get\_contact.collider\_velocity\_at\_pos** ( *int* contact\_idx ) const
- *float* **get\_step** ( ) const
- void **integrate\_forces** ( )
- *PhysicsDirectSpaceState* **get\_space\_state** ( )

## 9.208 PhysicsDirectBodyStateSW

**Inherits:** *PhysicsDirectBodyState* < *Object*

**Category:** Core

### 9.208.1 Brief Description

## 9.209 PhysicsDirectSpaceState

**Inherits:** *Object*

**Category:** Core

## 9.209.1 Brief Description

## 9.209.2 Member Functions

Dictionary	<code>intersect_ray ( Vector3 from, Vector3 to, Array exclude=Array(), int layer_mask=2147483647, int type_mask=15 )</code>
Array	<code>intersect_shape ( PhysicsShapeQueryParameters shape, int max_results=32 )</code>
Array	<code>cast_motion ( PhysicsShapeQueryParameters shape, Vector3 motion )</code>
Array	<code>collide_shape ( PhysicsShapeQueryParameters shape, int max_results=32 )</code>
Dictionary	<code>get_rest_info ( PhysicsShapeQueryParameters shape )</code>

## 9.209.3 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**

## 9.209.4 Member Function Description

- *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 )`
- *Array* `cast_motion ( PhysicsShapeQueryParameters shape, Vector3 motion )`
- *Array* `collide_shape ( PhysicsShapeQueryParameters shape, int max_results=32 )`
- *Dictionary* `get_rest_info ( PhysicsShapeQueryParameters shape )`

## 9.210 PhysicsServer

**Inherits:** *Object*

**Inherited By:** *PhysicsServerSW*

**Category:** Core

## 9.210.1 Brief Description

## 9.210.2 Member Functions

<code>RID</code>	<code>shape_create ( int type )</code>
	Continued on next page

Table 9.21 – continued from previous page

void	<code>shape_set_data ( RID shape, var data )</code>
<i>int</i>	<code>shape_get_type ( RID shape ) const</code>
void	<code>shape_get_data ( RID shape ) const</code>
<i>RID</i>	<code>space_create ()</code>
void	<code>space_set_active ( RID space, bool active )</code>
<i>bool</i>	<code>space_is_active ( RID space ) const</code>
void	<code>space_set_param ( RID space, int param, float value )</code>
<i>float</i>	<code>space_get_param ( RID space, int param ) const</code>
<i>PhysicsDirectSpaceState</i>	<code>space_get_direct_state ( RID space )</code>
<i>RID</i>	<code>area_create ()</code>
void	<code>area_set_space ( RID area, RID space )</code>
<i>RID</i>	<code>area_get_space ( RID area ) const</code>
void	<code>area_set_space_override_mode ( RID area, int mode )</code>
<i>int</i>	<code>area_get_space_override_mode ( RID area ) const</code>
void	<code>area_add_shape ( RID area, RID shape, Transform transform=Transform() )</code>
void	<code>area_set_shape ( RID area, int shape_idx, RID shape )</code>
void	<code>area_set_shape_transform ( RID area, int shape_idx, Transform transform )</code>
<i>int</i>	<code>area_get_shape_count ( RID area ) const</code>
<i>RID</i>	<code>area_get_shape ( RID area, int shape_idx ) const</code>
<i>Transform</i>	<code>area_get_shape_transform ( RID area, int shape_idx ) const</code>
void	<code>area_remove_shape ( RID area, int shape_idx )</code>
void	<code>area_clear_shapes ( RID area )</code>
void	<code>area_set_param ( RID area, int param, var value )</code>
void	<code>area_set_transform ( RID area, Transform transform )</code>
void	<code>area_get_param ( RID area, int param ) const</code>
<i>Transform</i>	<code>area_get_transform ( RID area ) const</code>
void	<code>area_attach_object_instance_ID ( RID area, int id )</code>
<i>int</i>	<code>area_get_object_instance_ID ( RID area ) const</code>
void	<code>area_set_monitor_callback ( RID area, Object receiver, String method )</code>
void	<code>area_set_ray_pickable ( RID area, bool enable )</code>
<i>bool</i>	<code>area_is_ray_pickable ( RID area ) const</code>
<i>RID</i>	<code>body_create ( int mode=2, bool init_sleeping=false )</code>
void	<code>body_set_space ( RID body, RID space )</code>
<i>RID</i>	<code>body_get_space ( RID body ) const</code>
void	<code>body_set_mode ( RID body, int mode )</code>
<i>int</i>	<code>body_get_mode ( RID body ) const</code>
void	<code>body_add_shape ( RID body, RID shape, Transform transform=Transform() )</code>
void	<code>body_set_shape ( RID body, int shape_idx, RID shape )</code>
void	<code>body_set_shape_transform ( RID body, int shape_idx, Transform transform )</code>
<i>int</i>	<code>body_get_shape_count ( RID body ) const</code>
<i>RID</i>	<code>body_get_shape ( RID body, int shape_idx ) const</code>
<i>Transform</i>	<code>body_get_shape_transform ( RID body, int shape_idx ) const</code>
void	<code>body_remove_shape ( RID body, int shape_idx )</code>
void	<code>body_clear_shapes ( RID body )</code>
void	<code>body_attach_object_instance_ID ( RID body, int id )</code>
<i>int</i>	<code>body_get_object_instance_ID ( RID body ) const</code>
void	<code>body_set_enable_continuous_collision_detection ( RID body, bool enable )</code>
<i>bool</i>	<code>body_is_continuous_collision_detection_enabled ( RID body ) const</code>
void	<code>body_set_param ( RID body, int param, float value )</code>
<i>float</i>	<code>body_get_param ( RID body, int param ) const</code>

Continued on next page

Table 9.21 – continued from previous page

void	<code>body_set_state ( RID body, int state, var value )</code>
void	<code>body_get_state ( RID body, int state ) const</code>
void	<code>body_apply_impulse ( RID body, Vector3 pos, Vector3 impulse )</code>
void	<code>body_set_axis_velocity ( RID body, Vector3 axis_velocity )</code>
void	<code>body_set_axis_lock ( RID body, int axis )</code>
<i>int</i>	<code>body_get_axis_lock ( RID body ) const</code>
void	<code>body_add_collision_exception ( RID body, RID excepted_body )</code>
void	<code>body_remove_collision_exception ( RID body, RID excepted_body )</code>
void	<code>body_set_max_contacts_reported ( RID body, int amount )</code>
<i>int</i>	<code>body_get_max_contacts_reported ( RID body ) const</code>
void	<code>body_set OMIT force_integration ( RID body, bool enable )</code>
<i>bool</i>	<code>body_is omitting force_integration ( RID body ) const</code>
void	<code>body_set_force_integration_callback ( RID body, Object receiver, String method, var userdata=NULL )</code>
void	<code>body_set_ray_pickable ( RID body, bool enable )</code>
<i>bool</i>	<code>body_is ray_pickable ( RID body ) const</code>
<i>RID</i>	<code>joint_create_pin ( RID body_A, Vector3 local_A, RID body_B, Vector3 local_B )</code>
void	<code>pin_joint_set_param ( RID joint, int param, float value )</code>
<i>float</i>	<code>pin_joint_get_param ( RID joint, int param ) const</code>
void	<code>pin_joint_set_local_A ( RID joint, Vector3 local_A )</code>
<i>Vector3</i>	<code>pin_joint_get_local_A ( RID joint ) const</code>
void	<code>pin_joint_set_local_B ( RID joint, Vector3 local_B )</code>
<i>Vector3</i>	<code>pin_joint_get_local_B ( RID joint ) const</code>
<i>RID</i>	<code>joint_create_hinge ( RID body_A, Transform hinge_A, RID body_B, Transform hinge_B )</code>
void	<code>hinge_joint_set_param ( RID joint, int param, float value )</code>
<i>float</i>	<code>hinge_joint_get_param ( RID joint, int param ) const</code>
void	<code>hinge_joint_set_flag ( RID joint, int flag, bool enabled )</code>
<i>bool</i>	<code>hinge_joint_get_flag ( RID joint, int flag ) const</code>
<i>RID</i>	<code>joint_create_slider ( RID body_A, Transform local_ref_A, RID body_B, Transform local_ref_B )</code>
void	<code>slider_joint_set_param ( RID joint, int param, float value )</code>
<i>float</i>	<code>slider_joint_get_param ( RID joint, int param ) const</code>
<i>RID</i>	<code>joint_create_cone_twist ( RID body_A, Transform local_ref_A, RID body_B, Transform local_ref_B )</code>
void	<code>cone_twist_joint_set_param ( RID joint, int param, float value )</code>
<i>float</i>	<code>cone_twist_joint_get_param ( RID joint, int param ) const</code>
<i>int</i>	<code>joint_get_type ( RID joint ) const</code>
void	<code>joint_set_solver_priority ( RID joint, int priority )</code>
<i>int</i>	<code>joint_get_solver_priority ( RID joint ) const</code>
<i>RID</i>	<code>joint_create_generic_6dof ( RID body_A, Transform local_ref_A, RID body_B, Transform local_ref_B )</code>
void	<code>generic_6dof_joint_set_param ( RID joint, int axis, int param, float value )</code>
<i>float</i>	<code>generic_6dof_joint_get_param ( RID joint, int axis, int param )</code>
void	<code>generic_6dof_joint_set_flag ( RID joint, int axis, int flag, bool enable )</code>
<i>bool</i>	<code>generic_6dof_joint_get_flag ( RID joint, int axis, int flag )</code>
void	<code>free_rid ( RID rid )</code>
void	<code>set_active ( bool active )</code>
<i>int</i>	<code>get_process_info ( int process_info )</code>

### 9.210.3 Numeric Constants

- **JOINT\_PIN = 0**
- **JOINTHINGE = 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**

## 9.210.4 Member Function Description

- `RID shape_create ( int type )`
- `void shape_set_data ( RID shape, var data )`
- `int shape_get_type ( RID shape ) const`
- `void shape_get_data ( RID shape ) const`
- `RID space_create ()`
- `void space_set_active ( RID space, bool active )`
- `bool space_is_active ( RID space ) const`
- `void space_set_param ( RID space, int param, float value )`
- `float space_get_param ( RID space, int param ) const`
- `PhysicsDirectSpaceState space_get_direct_state ( RID space )`
- `RID area_create ()`
- `void area_set_space ( RID area, RID space )`
- `RID area_get_space ( RID area ) const`
- `void area_set_space_override_mode ( RID area, int mode )`
- `int area_get_space_override_mode ( RID area ) const`
- `void area_add_shape ( RID area, RID shape, Transform transform=Transform() )`
- `void area_set_shape ( RID area, int shape_idx, RID shape )`
- `void area_set_shape_transform ( RID area, int shape_idx, Transform transform )`
- `int area_get_shape_count ( RID area ) const`
- `RID area_get_shape ( RID area, int shape_idx ) const`
- `Transform area_get_shape_transform ( RID area, int shape_idx ) const`
- `void area_remove_shape ( RID area, int shape_idx )`
- `void area_clear_shapes ( RID area )`
- `void area_set_param ( RID area, int param, var value )`
- `void area_set_transform ( RID area, Transform transform )`
- `void area_get_param ( RID area, int param ) const`
- `Transform area_get_transform ( RID area ) const`
- `void area_attach_object_instance_ID ( RID area, int id )`
- `int area_get_object_instance_ID ( RID area ) const`
- `void area_set_monitor_callback ( RID area, Object receiver, String method )`
- `void area_set_ray_pickable ( RID area, bool enable )`
- `bool area_is_ray_pickable ( RID area ) const`
- `RID body_create ( int mode=2, bool init_sleeping=false )`
- `void body_set_space ( RID body, RID space )`
- `RID body_get_space ( RID body ) const`

- void **body\_set\_mode** ( *RID* body, *int* mode )
- *int* **body\_get\_mode** ( *RID* body ) const
- void **body\_add\_shape** ( *RID* body, *RID* shape, *Transform* transform=Transform() )
- void **body\_set\_shape** ( *RID* body, *int* shape\_idx, *RID* shape )
- void **body\_set\_shape\_transform** ( *RID* body, *int* shape\_idx, *Transform* transform )
- *int* **body\_get\_shape\_count** ( *RID* body ) const
- *RID* **body\_get\_shape** ( *RID* body, *int* shape\_idx ) const
- *Transform* **body\_get\_shape\_transform** ( *RID* body, *int* shape\_idx ) const
- void **body\_remove\_shape** ( *RID* body, *int* shape\_idx )
- void **body\_clear\_shapes** ( *RID* body )
- void **body\_attach\_object\_instance\_ID** ( *RID* body, *int* id )
- *int* **body\_get\_object\_instance\_ID** ( *RID* body ) const
- void **body\_set\_enable\_continuous\_collision\_detection** ( *RID* body, *bool* enable )
- *bool* **body\_is\_continuous\_collision\_detection\_enabled** ( *RID* body ) const
- void **body\_set\_param** ( *RID* body, *int* param, *float* value )
- *float* **body\_get\_param** ( *RID* body, *int* param ) const
- void **body\_set\_state** ( *RID* body, *int* state, var value )
- void **body\_get\_state** ( *RID* body, *int* state ) const
- void **body\_apply\_impulse** ( *RID* body, *Vector3* pos, *Vector3* impulse )
- void **body\_set\_axis\_velocity** ( *RID* body, *Vector3* axis\_velocity )
- void **body\_set\_axis\_lock** ( *RID* body, *int* axis )
- *int* **body\_get\_axis\_lock** ( *RID* body ) const
- void **body\_add\_collision\_exception** ( *RID* body, *RID* excepted\_body )
- void **body\_remove\_collision\_exception** ( *RID* body, *RID* excepted\_body )
- void **body\_set\_max\_contacts\_reported** ( *RID* body, *int* amount )
- *int* **body\_get\_max\_contacts\_reported** ( *RID* body ) const
- void **body\_set OMIT force\_integration** ( *RID* body, *bool* enable )
- *bool* **body\_is omitting force\_integration** ( *RID* body ) const
- void **body\_set\_force\_integration\_callback** ( *RID* body, *Object* receiver, *String* method, var userdata=NULL )
- void **body\_set\_ray\_pickable** ( *RID* body, *bool* enable )
- *bool* **body\_is\_ray\_pickable** ( *RID* body ) const
- *RID* **joint\_create\_pin** ( *RID* body\_A, *Vector3* local\_A, *RID* body\_B, *Vector3* local\_B )
- void **pin\_joint\_set\_param** ( *RID* joint, *int* param, *float* value )
- *float* **pin\_joint\_get\_param** ( *RID* joint, *int* param ) const
- void **pin\_joint\_set\_local\_A** ( *RID* joint, *Vector3* local\_A )
- *Vector3* **pin\_joint\_get\_local\_A** ( *RID* joint ) const

- void **pin\_joint\_set\_local\_B** ( *RID* joint, *Vector3* local\_B )
- *Vector3* **pin\_joint\_get\_local\_B** ( *RID* joint ) const
- *RID* **joint\_create\_hinge** ( *RID* body\_A, *Transform* hinge\_A, *RID* body\_B, *Transform* hinge\_B )
- void **hinge\_joint\_set\_param** ( *RID* joint, *int* param, *float* value )
- *float* **hinge\_joint\_get\_param** ( *RID* joint, *int* param ) const
- void **hinge\_joint\_set\_flag** ( *RID* joint, *int* flag, *bool* enabled )
- *bool* **hinge\_joint\_get\_flag** ( *RID* joint, *int* flag ) const
- *RID* **joint\_create\_slider** ( *RID* body\_A, *Transform* local\_ref\_A, *RID* body\_B, *Transform* local\_ref\_B )
- void **slider\_joint\_set\_param** ( *RID* joint, *int* param, *float* value )
- *float* **slider\_joint\_get\_param** ( *RID* joint, *int* param ) const
- *RID* **joint\_create\_cone\_twist** ( *RID* body\_A, *Transform* local\_ref\_A, *RID* body\_B, *Transform* local\_ref\_B )
- void **cone\_twist\_joint\_set\_param** ( *RID* joint, *int* param, *float* value )
- *float* **cone\_twist\_joint\_get\_param** ( *RID* joint, *int* param ) const
- *int* **joint\_get\_type** ( *RID* joint ) const
- void **joint\_set\_solver\_priority** ( *RID* joint, *int* priority )
- *int* **joint\_get\_solver\_priority** ( *RID* joint ) const
- *RID* **joint\_create\_generic\_6dof** ( *RID* body\_A, *Transform* local\_ref\_A, *RID* body\_B, *Transform* local\_ref\_B )
- void **generic\_6dof\_joint\_set\_param** ( *RID* joint, *int* axis, *int* param, *float* value )
- *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 )
- *bool* **generic\_6dof\_joint\_get\_flag** ( *RID* joint, *int* axis, *int* flag )
- void **free\_rid** ( *RID* rid )
- void **set\_active** ( *bool* active )
- *int* **get\_process\_info** ( *int* process\_info )

## 9.211 PhysicsServerSW

Inherits: *PhysicsServer < Object*

Category: Core

### 9.211.1 Brief Description

## 9.212 PhysicsShapeQueryParameters

Inherits: *Reference < Object*

Category: Core

## 9.212.1 Brief Description

## 9.212.2 Member Functions

void	<code>set_shape ( Shape shape )</code>
void	<code>set_shape_rid ( RID shape )</code>
<i>RID</i>	<code>get_shape_rid () const</code>
void	<code>set_transform ( Transform transform )</code>
<i>Transform</i>	<code>get_transform () const</code>
void	<code>set_margin ( float margin )</code>
<i>float</i>	<code>get_margin () const</code>
void	<code>set_layer_mask ( int layer_mask )</code>
<i>int</i>	<code>get_layer_mask () const</code>
void	<code>set_object_type_mask ( int object_type_mask )</code>
<i>int</i>	<code>get_object_type_mask () const</code>
void	<code>set_exclude ( Array exclude )</code>
<i>Array</i>	<code>get_exclude () const</code>

## 9.212.3 Member Function Description

- void `set_shape ( Shape shape )`
- void `set_shape_rid ( RID shape )`
- *RID* `get_shape_rid () const`
- void `set_transform ( Transform transform )`
- *Transform* `get_transform () const`
- void `set_margin ( float margin )`
- *float* `get_margin () const`
- void `set_layer_mask ( int layer_mask )`
- *int* `get_layer_mask () const`
- void `set_object_type_mask ( int object_type_mask )`
- *int* `get_object_type_mask () const`
- void `set_exclude ( Array exclude )`
- *Array* `get_exclude () const`

## 9.213 PhysicsShapeQueryResult

Inherits: [Reference < Object](#)

Category: Core

### 9.213.1 Brief Description

Result of a shape query in Physics2DServer.

### 9.213.2 Member Functions

<i>int</i>	<code>get_result_count () const</code>
<i>RID</i>	<code>get_result_rid ( int idx ) const</code>
<i>int</i>	<code>get_result_object_id ( int idx ) const</code>
<i>Object</i>	<code>get_result_object ( int idx ) const</code>
<i>int</i>	<code>get_result_object_shape ( int idx ) const</code>

### 9.213.3 Member Function Description

- `int get_result_count () const`
- `RID get_result_rid ( int idx ) const`
- `int get_result_object_id ( int idx ) const`
- `Object get_result_object ( int idx ) const`
- `int get_result_object_shape ( int idx ) const`

## 9.214 PinJoint

Inherits: *Joint* < *Spatial* < *Node* < *Object*

Category: Core

### 9.214.1 Brief Description

### 9.214.2 Member Functions

<code>void</code>	<code>set_param ( int param, float value )</code>
<code>float</code>	<code>get_param ( int param ) const</code>

### 9.214.3 Numeric Constants

- `PARAM_BIAS = 0`
- `PARAM_DAMPING = 1`
- `PARAM_IMPULSE_CLAMP = 2`

### 9.214.4 Member Function Description

- `void set_param ( int param, float value )`
- `float get_param ( int param ) const`

## 9.215 PinJoint2D

**Inherits:** *Joint2D < Node2D < CanvasItem < Node < Object*

**Category:** Core

### 9.215.1 Brief Description

Pin Joint for 2D Shapes.

### 9.215.2 Member Functions

void	<code>set_softness (float softness)</code>
<code>float</code>	<code>get_softness () const</code>

### 9.215.3 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.

### 9.215.4 Member Function Description

- void `set_softness (float softness)`
- `float get_softness () const`

## 9.216 Plane

**Category:** Built-In Types

### 9.216.1 Brief Description

Plane in hessian form.

## 9.216.2 Member Functions

<i>Vector3</i>	<code>center ()</code>
<i>float</i>	<code>distance_to ( Vector3 point )</code>
<i>Vector3</i>	<code>get_any_point ()</code>
<i>bool</i>	<code>has_point ( Vector3 point, float epsilon=0.00001 )</code>
<i>Vector3</i>	<code>intersect_3 ( Plane b, Plane c )</code>
<i>Vector3</i>	<code>intersects_ray ( Vector3 from, Vector3 dir )</code>
<i>Vector3</i>	<code>intersects_segment ( Vector3 begin, Vector3 end )</code>
<i>bool</i>	<code>is_point_over ( Vector3 point )</code>
<i>Plane</i>	<code>normalized ()</code>
<i>Vector3</i>	<code>project ( Vector3 point )</code>
<i>Plane</i>	<code>Plane ( float a, float b, float c, float d )</code>
<i>Plane</i>	<code>Plane ( Vector3 v1, Vector3 v2, Vector3 v3 )</code>
<i>Plane</i>	<code>Plane ( Vector3 normal, float d )</code>

## 9.216.3 Member Variables

- *Vector3* **normal**
- *float* **x**
- *float* **y**
- *float* **z**
- *float* **d**

## 9.216.4 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.

## 9.216.5 Member Function Description

- *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.

- `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.

## 9.217 PlaneShape

**Inherits:** `Shape < Resource < Reference < Object`

**Category:** Core

### 9.217.1 Brief Description

### 9.217.2 Member Functions

void	<code>set_plane ( Plane plane )</code>
<code>Plane</code>	<code>get_plane () const</code>

### 9.217.3 Member Function Description

- `void set_plane ( Plane plane )`
- `Plane get_plane () const`

## 9.218 Polygon2D

**Inherits:** `Node2D < CanvasItem < Node < Object`

**Category:** Core

### 9.218.1 Brief Description

### 9.218.2 Member Functions

void	<code>set_polygon ( Vector2Array polygon )</code>
<code>Vector2Array</code>	<code>get_polygon () const</code>
void	<code>set_uv ( Vector2Array uv )</code>
<code>Vector2Array</code>	<code>get_uv () const</code>
void	<code>set_color ( Color color )</code>
<code>Color</code>	<code>get_color () const</code>
void	<code>set_texture ( Object texture )</code>
<code>Object</code>	<code>get_texture () const</code>
void	<code>set_texture_offset ( Vector2 texture_offset )</code>
<code>Vector2</code>	<code>get_texture_offset () const</code>
void	<code>set_texture_rotation ( float texture_rotation )</code>
<code>float</code>	<code>get_texture_rotation () const</code>
void	<code>set_texture_scale ( Vector2 texture_scale )</code>
<code>Vector2</code>	<code>get_texture_scale () const</code>
void	<code>set_invert ( bool invert )</code>
<code>bool</code>	<code>get_invert () const</code>
void	<code>set_invert_border ( float invert_border )</code>
<code>float</code>	<code>get_invert_border () const</code>
void	<code>set_offset ( Vector2 offset )</code>
<code>Vector2</code>	<code>get_offset () const</code>

### 9.218.3 Member Function Description

- void `set_polygon ( Vector2Array polygon )`
- `Vector2Array` `get_polygon () const`
- void `set_uv ( Vector2Array uv )`
- `Vector2Array` `get_uv () const`
- void `set_color ( Color color )`
- `Color` `get_color () const`
- void `set_texture ( Object texture )`
- `Object` `get_texture () const`
- void `set_texture_offset ( Vector2 texture_offset )`
- `Vector2` `get_texture_offset () const`
- void `set_texture_rotation ( float texture_rotation )`
- `float` `get_texture_rotation () const`
- void `set_texture_scale ( Vector2 texture_scale )`
- `Vector2` `get_texture_scale () const`
- void `set_invert ( bool invert )`
- `bool` `get_invert () const`
- void `set_invert_border ( float invert_border )`

- `float get_invert_border () const`
- `void set_offset ( Vector2 offset )`
- `Vector2 get_offset () const`

## 9.219 PolygonPathFinder

**Inherits:** `Resource < Reference < Object`

**Category:** Core

### 9.219.1 Brief Description

### 9.219.2 Member Functions

<code>void</code>	<code>setup ( Vector2Array points, IntArray connections )</code>
<code>Vector2Array</code>	<code>find_path ( Vector2 from, Vector2 to )</code>
<code>Vector2Array</code>	<code>get_intersections ( Vector2 from, Vector2 to ) const</code>
<code>Vector2</code>	<code>get_closest_point ( Vector2 point ) const</code>
<code>bool</code>	<code>is_point_inside ( Vector2 point ) const</code>
<code>void</code>	<code>set_point_penalty ( int idx, float penalty )</code>
<code>float</code>	<code>get_point_penalty ( int idx ) const</code>
<code>Rect2</code>	<code>get_bounds () const</code>

### 9.219.3 Member Function Description

- `void setup ( Vector2Array points, IntArray connections )`
- `Vector2Array find_path ( Vector2 from, Vector2 to )`
- `Vector2Array get_intersections ( Vector2 from, Vector2 to ) const`
- `Vector2 get_closest_point ( Vector2 point ) const`
- `bool is_point_inside ( Vector2 point ) const`
- `void set_point_penalty ( int idx, float penalty )`
- `float get_point_penalty ( int idx ) const`
- `Rect2 get_bounds () const`

## 9.220 Popup

**Inherits:** `Control < CanvasItem < Node < Object`

**Inherited By:** `PopupPanel, PopupDialog, PopupMenu, WindowDialog`

**Category:** Core

### 9.220.1 Brief Description

Base container control for popups and dialogs.

## 9.220.2 Member Functions

void	<code>popup_centered ( Vector2 size=Vector2(0,0) )</code>
void	<code>popup_centered_ratio ( float ratio=0.75 )</code>
void	<code>popup_centered_minsize ( Vector2 minsize=Vector2(0,0) )</code>
void	<code>popup ()</code>
void	<code>set_exclusive ( bool enable )</code>
<code>bool</code>	<code>is_exclusive ( ) const</code>

## 9.220.3 Signals

- `popup_hide ()`
- `about_to_show ()`

## 9.220.4 Numeric Constants

- `NOTIFICATION_POST_POPUP = 80`
- `NOTIFICATION_POPUP_HIDE = 81`

## 9.220.5 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.

## 9.220.6 Member Function Description

- 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_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 `popup_centered_minsize ( Vector2 minsize=Vector2(0,0) )`

- void `popup ()`

Popup (show the control in modal form).

- void `set_exclusive ( bool enable )`
- `bool is_exclusive ( ) const`

## 9.221 PopupDialog

**Inherits:** `Popup < Control < CanvasItem < Node < Object`

**Category:** Core

## 9.221.1 Brief Description

Base class for Popup Dialogs.

## 9.222 PopupMenu

**Inherits:** [Popup](#) < [Control](#) < [CanvasItem](#) < [Node](#) < [Object](#)

**Category:** Core

### 9.222.1 Brief Description

PopupMenu displays a list of options.

### 9.222.2 Member Functions

void	<code>add_icon_item ( Object texture, String label, int id=-1, int accel=0 )</code>
void	<code>add_item ( String label, int id=-1, int accel=0 )</code>
void	<code>add_icon_check_item ( Object texture, String label, int id=-1, int accel=0 )</code>
void	<code>add_check_item ( String label, int id=-1, int accel=0 )</code>
void	<code>add_submenu_item ( String label, String submenu, int id=-1 )</code>
void	<code>set_item_text ( int idx, String text )</code>
void	<code>set_item_icon ( int idx, Object icon )</code>
void	<code>set_item_accelerator ( int idx, int accel )</code>
void	<code>set_item_metadata ( int idx, var metadata )</code>
void	<code>set_item_checked ( int idx, bool checked )</code>
void	<code>set_item_disabled ( int idx, bool disabled )</code>
void	<code>set_item_submenu ( int idx, String submenu )</code>
void	<code>set_item_as_separator ( int idx, bool enable )</code>
void	<code>set_item_as_checkable ( int idx, bool enable )</code>
void	<code>set_item_ID ( int idx, int id )</code>
<i>String</i>	<code>get_item_text ( int idx ) const</code>
<i>Object</i>	<code>get_item_icon ( int idx ) const</code>
void	<code>get_item_metadata ( int idx ) const</code>
<i>int</i>	<code>get_item_accelerator ( int idx ) const</code>
<i>String</i>	<code>get_item_submenu ( int idx ) const</code>
<i>bool</i>	<code>is_item_separator ( int idx ) const</code>
<i>bool</i>	<code>is_item_checkable ( int idx ) const</code>
<i>bool</i>	<code>is_item_checked ( int idx ) const</code>
<i>bool</i>	<code>is_item_disabled ( int idx ) const</code>
<i>int</i>	<code>get_item_ID ( int idx ) const</code>
<i>int</i>	<code>get_item_index ( int id ) const</code>
<i>int</i>	<code>get_item_count () const</code>
void	<code>add_separator ()</code>
void	<code>remove_item ( int idx )</code>
void	<code>clear ()</code>

### 9.222.3 Signals

- **item\_pressed** ( *int* ID )

### 9.222.4 Description

PopupMenu is the typical Control that displays a list of options. They are popular in toolbars or context menus.

### 9.222.5 Member Function Description

- 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. If no id is provided, one will be created from the index.

- void **add\_item** ( *String* label, *int* id=-1, *int* accel=0 )

Add a new 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.

- 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\_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\_submenu\_item** ( *String* label, *String* submenu, *int* id=-1 )

- void **set\_item\_text** ( *int* idx, *String* text )

Set the text of the item at index “idx”.

- void **set\_item\_icon** ( *int* idx, *Object* icon )

Set the icon 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\_metadata** ( *int* idx, var metadata )

- 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 )

- void **set\_item\_submenu** ( *int* idx, *String* submenu )

- void **set\_item\_as\_separator** ( *int* idx, *bool* enable )

- void **set\_item\_as\_checkable** ( *int* idx, *bool* enable )

- void **set\_item\_ID** ( *int* idx, *int* id )

Set the id of the item at index “idx”.

- `String get_item_text ( int idx ) const`

Return the text of the item at index “idx”.

- `Object get_item_icon ( int idx ) const`

Return the icon of the item at index “idx”.

- `void get_item_metadata ( int idx ) const`
- `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.

- `String get_item_submenu ( int idx ) const`
- `bool is_item_separator ( int idx ) const`
- `bool is_item_checkable ( int idx ) const`
- `bool is_item_checked ( int idx ) const`

Return the checkstate status of the item at index “idx”.

- `bool is_item_disabled ( int idx ) const`
- `int get_item_ID ( int idx ) const`

Return the id 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.

- `int get_item_count ( ) const`

Return the amount of items.

- `void add_separator ( )`

Add a separator between items. Separators also occupy an index.

- `void remove_item ( int idx )`
- `void clear ( )`

Clear the popup menu.

## 9.223 PopupPanel

**Inherits:** `Popup < Control < CanvasItem < Node < Object`

**Category:** Core

### 9.223.1 Brief Description

Base class for Popup Panels

## 9.224 Portal

**Inherits:** *VisualInstance < Spatial < Node < Object*

**Category:** Core

### 9.224.1 Brief Description

Portals provide virtual openings to rooms.

### 9.224.2 Member Functions

void	<i>set_shape ( Vector2Array points )</i>
<i>Vector2Array</i>	<i>get_shape () const</i>
void	<i>set_enabled ( bool enable )</i>
<i>bool</i>	<i>is_enabled () const</i>
void	<i>set_disable_distance ( float distance )</i>
<i>float</i>	<i>get_disable_distance () const</i>
void	<i>set_disabled_color ( Color color )</i>
<i>Color</i>	<i>get_disabled_color () const</i>
void	<i>set_connect_range ( float range )</i>
<i>float</i>	<i>get_connect_range () const</i>

### 9.224.3 Description

Portals provide virtual openings to RoomInstance 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 RoomInstance. 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 RoomInstance, 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 RoomInstance node as a parent, grandparent or far parent, or else they will not be active.

### 9.224.4 Member Function Description

- void **set\_shape** ( *Vector2Array* points )

Set the portal shape. The shape is an array of Point2 points, representing a convex polygon in the X,Y plane.

- *Vector2Array* **get\_shape** ( ) const

Return the portal shape. The shape is an array of Point2 points, representing a convex polygon in the X,Y plane.

- void **set\_enabled** ( *bool* enable )

Enable the portal (it is enabled by default though), disabling it will cause the parent RoomInstance to not be visible any longer when looking through the portal.

- *bool* **is\_enabled** ( ) const

Return whether the portal is active. When disabled it causes the parent RoomInstance to not be visible any longer when looking through the portal.

- `void set_disable_distance (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`).

- `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`).

- `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”.

- `Color get_disabled_color () const`

Return the color for when the portal goes beyond the disable distance (see `set_disable_distance`) and becomes disabled.

- `void set_connect_range (float range)`

Set the range for auto-connecting two portals from different rooms sharing the same space.

- `float get_connect_range () const`

Return the range for auto-connecting two portals from different rooms sharing the same space.

## 9.225 Position2D

**Inherits:** `Node2D < CanvasItem < Node < Object`

**Category:** Core

### 9.225.1 Brief Description

Generic 2D Position hint for editing.

### 9.225.2 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.

## 9.226 Position3D

**Inherits:** `Spatial < Node < Object`

**Category:** Core

### 9.226.1 Brief Description

Generic 3D Position hint for editing

## 9.226.2 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.

## 9.227 ProgressBar

**Inherits:** *Range* < *Control* < *CanvasItem* < *Node* < *Object*

**Category:** Core

### 9.227.1 Brief Description

General purpose progress bar.

### 9.227.2 Member Functions

void	<code>set_percent_visible ( bool visible )</code>
<code>bool</code>	<code>is_percent_visible ( ) const</code>

### 9.227.3 Description

General purpose progress bar. Shows fill percentage from right to left.

### 9.227.4 Member Function Description

- void `set_percent_visible ( bool visible )`
- `bool is_percent_visible ( ) const`

## 9.228 ProximityGroup

**Inherits:** *Spatial* < *Node* < *Object*

**Category:** Core

### 9.228.1 Brief Description

General purpose proximity-detection node.

### 9.228.2 Member Functions

void	<code>set_group_name ( String name )</code>
void	<code>broadcast ( String name, var parameters )</code>
void	<code>set_dispatch_mode ( int mode )</code>
void	<code>set_grid_radius ( Vector3 radius )</code>
<code>Vector3</code>	<code>get_grid_radius ( ) const</code>

### 9.228.3 Signals

- **broadcast** ( *String* name, *Array* parameters )

### 9.228.4 Description

General purpose proximity-detection node.

### 9.228.5 Member Function Description

- void **set\_group\_name** ( *String* name )
- void **broadcast** ( *String* name, var parameters )
- void **set\_dispatch\_mode** ( *int* mode )
- void **set\_grid\_radius** ( *Vector3* radius )
- *Vector3* **get\_grid\_radius** ( ) const

## 9.229 Quad

**Inherits:** *GeometryInstance* < *VisualInstance* < *Spatial* < *Node* < *Object*

**Category:** Core

### 9.229.1 Brief Description

### 9.229.2 Member Functions

void	<i>set_axis</i> ( <i>int</i> axis )
<i>int</i>	<i>get_axis</i> ( ) const
void	<i>set_size</i> ( <i>Vector2</i> size )
<i>Vector2</i>	<i>get_size</i> ( ) const
void	<i>set_centered</i> ( <i>bool</i> centered )
<i>bool</i>	<i>is_centered</i> ( ) const
void	<i>set_offset</i> ( <i>Vector2</i> offset )
<i>Vector2</i>	<i>get_offset</i> ( ) const

### 9.229.3 Member Function Description

- void **set\_axis** ( *int* axis )
- *int* **get\_axis** ( ) const
- void **set\_size** ( *Vector2* size )
- *Vector2* **get\_size** ( ) const
- void **set\_centered** ( *bool* centered )
- *bool* **is\_centered** ( ) const
- void **set\_offset** ( *Vector2* offset )

- *Vector2* `get_offset()` const

## 9.230 Quat

**Category:** Built-In Types

### 9.230.1 Brief Description

Quaternion.

### 9.230.2 Member Functions

<i>Quat</i>	<code>cubic_slerp ( Quat b, Quat pre_a, Quat post_b, float t )</code>
<i>float</i>	<code>dot ( Quat b )</code>
<i>Quat</i>	<code>inverse ()</code>
<i>float</i>	<code>length ()</code>
<i>float</i>	<code>length_squared ()</code>
<i>Quat</i>	<code>normalized ()</code>
<i>Quat</i>	<code>slerp ( Quat b, float t )</code>
<i>Quat</i>	<code>slerpni ( Quat b, float t )</code>
<i>Vector3</i>	<code>xform ( Vector3 v )</code>
<i>Quat</i>	<code>Quat ( float x, float y, float z, float w )</code>
<i>Quat</i>	<code>Quat ( Vector3 axis, float angle )</code>
<i>Quat</i>	<code>Quat ( Matrix3 from )</code>

### 9.230.3 Member Variables

- *float* `x`
- *float* `y`
- *float* `z`
- *float* `w`

### 9.230.4 Description

Quaternion is a 4 dimensional vector that is used to represent a rotation. It mainly exists to perform SLERP (spherical-linear interpolation) between two 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.

### 9.230.5 Member Function Description

- *Quat* `cubic_slerp ( Quat b, Quat pre_a, Quat post_b, float t )`
- *float* `dot ( Quat b )`

Returns the dot product between two quaternions.

- *Quat* `inverse ()`

Returns the inverse of the quaternion (applies to the inverse rotation too).

- `float length ()`

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 )`
- `Quat Quat ( float x, float y, float z, float w )`
- `Quat Quat ( Vector3 axis, float angle )`
- `Quat Quat ( Matrix3 from )`

## 9.231 Range

**Inherits:** `Control < CanvasItem < Node < Object`

**Inherited By:** `SpinBox, ScrollBar, ProgressBar, TextureProgress, Slider`

**Category:** Core

### 9.231.1 Brief Description

Abstract base class for range-based controls.

## 9.231.2 Member Functions

<code>float</code>	<code>get_val()</code> const
<code>float</code>	<code>get_value()</code> const
<code>float</code>	<code>get_min()</code> const
<code>float</code>	<code>get_max()</code> const
<code>float</code>	<code>get_step()</code> const
<code>float</code>	<code>get_page()</code> const
<code>float</code>	<code>get_unit_value()</code> const
<code>void</code>	<code>set_val(float value)</code>
<code>void</code>	<code>set_value(float value)</code>
<code>void</code>	<code>set_min(float minimum)</code>
<code>void</code>	<code>set_max(float maximum)</code>
<code>void</code>	<code>set_step(float step)</code>
<code>void</code>	<code>set_page(float pagesize)</code>
<code>void</code>	<code>set_unit_value(float value)</code>
<code>void</code>	<code>set_rounded_values(bool enabled)</code>
<code>bool</code>	<code>is_rounded_values()</code> const
<code>void</code>	<code>set_exp_unit_value(bool enabled)</code>
<code>bool</code>	<code>is_unit_value_exp()</code> const
<code>void</code>	<code>share(Object with)</code>
<code>void</code>	<code>unshare()</code>

## 9.231.3 Signals

- `value_changed(float value)`
- `changed()`

## 9.231.4 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*.

## 9.231.5 Member Function Description

- `float get_val()` const

Return the current value.

- `float get_value()` const
- `float get_min()` const

Return the minimum value.

- `float get_max()` const

Return the maximum value.

- `float get_step()` const

Return the stepping, if step is 0, stepping is disabled.

- `float get_page()` const

Return the page size, if page is 0, paging is disabled.

- `float get_unit_value () const`

Return value mapped to 0 to 1 (unit) range.

- `void set_val (float value)`
- `void set_value (float value)`
- `void set_min (float minimum)`

Set minimum value, clamped range value to it if it's less.

- `void set_max (float maximum)`
- `void set_step (float step)`

Set step value. If step is 0, stepping will be disabled.

- `void set_page (float pagesize)`

Set page size. Page is mainly used for scrollbars or anything that controls text scrolling.

- `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_rounded_values (bool enabled)`
- `bool is_rounded_values () const`
- `void set_exp_unit_value (bool enabled)`
- `bool is_unit_value_exp () const`
- `void share (Object with)`
- `void unshare ()`

## 9.232 RawArray

**Category:** Built-In Types

### 9.232.1 Brief Description

Raw byte array.

### 9.232.2 Member Functions

<code>String</code>	<code>get_string_from_ascii ()</code>
<code>String</code>	<code>get_string_from_utf8 ()</code>
<code>void</code>	<code>push_back (<i>int</i> byte)</code>
<code>void</code>	<code>resize (<i>int</i> idx)</code>
<code>void</code>	<code>set (<i>int</i> idx, <i>int</i> byte)</code>
<code>int</code>	<code>size ()</code>
<code>RawArray</code>	<code>RawArray (<i>Array</i> from)</code>

### 9.232.3 Description

Raw byte array. Contains bytes. Optimized for memory usage, can't fragment the memory.

### 9.232.4 Member Function Description

- `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.

- `void push_back (int byte)`
- `void resize (int idx)`
- `void set (int idx, int byte)`
- `int size()`
- `RawArray RawArray (Array from)`

## 9.233 RayCast

**Inherits:** `Spatial < Node < Object`

**Category:** Core

### 9.233.1 Brief Description

### 9.233.2 Member Functions

<code>void</code>	<code>set_enabled (bool enabled)</code>
<code>bool</code>	<code>is_enabled () const</code>
<code>void</code>	<code>set_cast_to (Vector3 local_point)</code>
<code>Vector3</code>	<code>get_cast_to () const</code>
<code>bool</code>	<code>is_colliding () const</code>
<code>Object</code>	<code>get_collider () const</code>
<code>int</code>	<code>get_collider_shape () const</code>
<code>Vector3</code>	<code>get_collision_point () const</code>
<code>Vector3</code>	<code>get_collision_normal () const</code>
<code>void</code>	<code>add_exception_rid (RID rid)</code>
<code>void</code>	<code>add_exception (Object node)</code>
<code>void</code>	<code>remove_exception_rid (RID rid)</code>
<code>void</code>	<code>remove_exception (Object node)</code>
<code>void</code>	<code>clear_exceptions ()</code>

### 9.233.3 Member Function Description

- void **set\_enabled** ( *bool* enabled )
- *bool* **is\_enabled** ( ) const
- void **set\_cast\_to** ( *Vector3* local\_point )
- *Vector3* **get\_cast\_to** ( ) const
- *bool* **is\_colliding** ( ) const
- *Object* **get\_collider** ( ) const
- *int* **get\_collider\_shape** ( ) const
- *Vector3* **get\_collision\_point** ( ) const
- *Vector3* **get\_collision\_normal** ( ) const
- void **add\_exception\_rid** ( *RID* rid )
- void **add\_exception** ( *Object* node )
- void **remove\_exception\_rid** ( *RID* rid )
- void **remove\_exception** ( *Object* node )
- void **clear\_exceptions** ( )

## 9.234 RayCast2D

**Inherits:** *Node2D* < *CanvasItem* < *Node* < *Object*

**Category:** Core

### 9.234.1 Brief Description

### 9.234.2 Member Functions

void	<i>set_enabled</i> ( <i>bool</i> enabled )
<i>bool</i>	<i>is_enabled</i> ( ) const
void	<i>set_cast_to</i> ( <i>Vector2</i> local_point )
<i>Vector2</i>	<i>get_cast_to</i> ( ) const
<i>bool</i>	<i>is_colliding</i> ( ) const
<i>Object</i>	<i>get_collider</i> ( ) const
<i>int</i>	<i>get_collider_shape</i> ( ) const
<i>Vector2</i>	<i>get_collision_point</i> ( ) const
<i>Vector2</i>	<i>get_collision_normal</i> ( ) const
void	<i>add_exception_rid</i> ( <i>RID</i> rid )
void	<i>add_exception</i> ( <i>Object</i> node )
void	<i>remove_exception_rid</i> ( <i>RID</i> rid )
void	<i>remove_exception</i> ( <i>Object</i> node )
void	<i>clear_exceptions</i> ( )
void	<i>set_layer_mask</i> ( <i>int</i> mask )
<i>int</i>	<i>get_layer_mask</i> ( ) const
void	<i>set_type_mask</i> ( <i>int</i> mask )
<i>int</i>	<i>get_type_mask</i> ( ) const

### 9.234.3 Member Function Description

- void **set\_enabled** ( *bool* enabled )
- *bool* **is\_enabled** ( ) const
- void **set\_cast\_to** ( *Vector2* local\_point )
- *Vector2* **get\_cast\_to** ( ) const
- *bool* **is\_colliding** ( ) const

Return whether the closest object the ray is pointing to is colliding with the vector, with the vector length considered.

- *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 *is\_colliding* to check if the object returned is actually colliding with the ray.

- *int* **get\_collider\_shape** ( ) const
- *Vector2* **get\_collision\_point** ( ) const
- *Vector2* **get\_collision\_normal** ( ) const
- void **add\_exception\_rid** ( *RID* rid )
- void **add\_exception** ( *Object* node )
- void **remove\_exception\_rid** ( *RID* rid )
- void **remove\_exception** ( *Object* node )
- void **clear\_exceptions** ( )
- void **set\_layer\_mask** ( *int* mask )
- *int* **get\_layer\_mask** ( ) const
- void **set\_type\_mask** ( *int* mask )
- *int* **get\_type\_mask** ( ) const

## 9.235 RayShape

**Inherits:** *Shape* < *Resource* < *Reference* < *Object*

**Category:** Core

### 9.235.1 Brief Description

### 9.235.2 Member Functions

<i>void</i>	<i>set_length</i> ( <i>float</i> length )
<i>float</i>	<i>get_length</i> ( ) const

### 9.235.3 Member Function Description

- void `set_length (float length)`
- `float get_length () const`

## 9.236 RayShape2D

Inherits: [Shape2D](#) < [Resource](#) < [Reference](#) < [Object](#)

Category: Core

### 9.236.1 Brief Description

Ray 2D shape resource for physics.

### 9.236.2 Member Functions

void	<code>set_length (float length)</code>
<code>float</code>	<code>get_length () const</code>

### 9.236.3 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.

### 9.236.4 Member Function Description

- void `set_length (float length)`

Set the length of the ray.

- `float get_length () const`

Return the length of the ray.

## 9.237 RealArray

Category: Built-In Types

### 9.237.1 Brief Description

Real Array .

## 9.237.2 Member Functions

void	<i>push_back</i> ( <i>float</i> value )
void	<i>resize</i> ( <i>int</i> idx )
void	<i>set</i> ( <i>int</i> idx, <i>float</i> value )
<i>int</i>	<i>size</i> ()
<i>RealArray</i>	<i>RealArray</i> ( <i>Array</i> from )

## 9.237.3 Description

Real Array. Array of floating point values. Can only contain floats. Optimized for memory usage, can't fragment the memory.

## 9.237.4 Member Function Description

- void **push\_back** ( *float* value )
- void **resize** ( *int* idx )
- void **set** ( *int* idx, *float* value )
- *int* **size** ()
- *RealArray* **RealArray** ( *Array* from )

## 9.238 Rect2

**Category:** Built-In Types

### 9.238.1 Brief Description

2D Axis-aligned bounding box.

### 9.238.2 Member Functions

<i>Rect2</i>	<i>clip</i> ( <i>Rect2</i> b )
<i>bool</i>	<i>encloses</i> ( <i>Rect2</i> b )
<i>Rect2</i>	<i>expand</i> ( <i>Vector2</i> to )
<i>float</i>	<i>get_area</i> ()
<i>Rect2</i>	<i>grow</i> ( <i>float</i> by )
<i>bool</i>	<i>has_no_area</i> ()
<i>bool</i>	<i>has_point</i> ( <i>Vector2</i> point )
<i>bool</i>	<i>intersects</i> ( <i>Rect2</i> b )
<i>Rect2</i>	<i>merge</i> ( <i>Rect2</i> b )
<i>Rect2</i>	<i>Rect2</i> ( <i>Vector2</i> pos, <i>Vector2</i> size )
<i>Rect2</i>	<i>Rect2</i> ( <i>float</i> x, <i>float</i> y, <i>float</i> width, <i>float</i> height )

### 9.238.3 Member Variables

- *Vector2* **pos**
- *Vector2* **size**
- *Vector2* **end**

### 9.238.4 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.

### 9.238.5 Member Function Description

- *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.

- *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.

- *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.

## 9.239 RectangleShape2D

**Inherits:** [Shape2D](#) < [Resource](#) < [Reference](#) < [Object](#)

**Category:** Core

### 9.239.1 Brief Description

Rectangle Shape for 2D Physics.

### 9.239.2 Member Functions

void	<a href="#"><code>set_extents ( Vector2 extents )</code></a>
<code>Vector2</code>	<a href="#"><code>get_extents () const</code></a>

### 9.239.3 Description

Rectangle Shape for 2D Physics. This shape is useful for modeling box-like 2D objects.

### 9.239.4 Member Function Description

- void [`set\_extents \( Vector2 extents \)`](#)

Set the half extents, the actual width and height of this shape is twice the half extents.

- `Vector2 get_extents () const`

Return the half extents, the actual width and height of this shape is twice the half extents.

## 9.240 Reference

**Inherits:** [Object](#)

**Inherited By:** [RegEx](#), [SurfaceTool](#), [EditorScenePostImport](#), [PhysicsShapeQueryResult](#), [Physics2DTestMotionResult](#), [FuncRef](#), [File](#), [TCP\\_Server](#), [Physics2DShapeQueryResult](#), [ConfigFile](#), [StreamPeer](#), [HTTPClient](#), [AudioStreamPlayback](#), [MeshDataTool](#), [GDFunctionState](#), [Physics2DShapeQueryParameters](#), [EditorScript](#), [Mutex](#), [PacketPeer](#), [Semaphore](#), [XMLParser](#), [EditorImportPlugin](#), [Directory](#), [Marshalls](#), [WeakRef](#), [SceneState](#), [GDNativeClass](#), [PCK-Packer](#), [Resource](#), [Thread](#), [PackedDataContainerRef](#), [ResourceInteractiveLoader](#), [ResourceImportMetadata](#), [PhysicsShapeQueryParameters](#)

**Category:** Core

### 9.240.1 Brief Description

Base class for anything that keeps a reference count.

## 9.240.2 Member Functions

<i>bool</i>	<a href="#">init_ref ()</a>
<i>void</i>	<a href="#">reference ()</a>
<i>bool</i>	<a href="#">unreference ()</a>

## 9.240.3 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.

## 9.240.4 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.

## 9.241 ReferenceFrame

**Inherits:** [Control](#) < [CanvasItem](#) < [Node](#) < [Object](#)

**Category:** Core

### 9.241.1 Brief Description

Reference frame for GUI.

### 9.241.2 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.

## 9.242 RegEx

**Inherits:** [Reference](#) < [Object](#)

**Category:** Core

### 9.242.1 Brief Description

Simple regular expression matcher.

## 9.242.2 Member Functions

<code>int</code>	<code>compile ( String pattern, int capture=9 )</code>
<code>int</code>	<code>find ( String text, int start=0, int end=-1 ) const</code>
<code>void</code>	<code>clear ()</code>
<code>bool</code>	<code>is_valid () const</code>
<code>int</code>	<code>get_capture_count () const</code>
<code>String</code>	<code>get_capture ( int capture ) const</code>
<code>StringArray</code>	<code>get_captures () const</code>

## 9.242.3 Description

Class for finding text patterns in a string using regular expressions. Regular expressions are a way to define patterns of text to be searched.

This class only finds patterns in a string. It can not perform replacements.

Usage of regular expressions is too long to be explained here, but Internet is full of tutorials and detailed explanations.

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 :ref:`A-Za-z<class\_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 :ref:`[:alnum:<class\_[alnum]>:]`

Lookahead (?=), (?!) and lookbehind (?<=), (?<!)

ASCII \xFF and Unicode \uFFFF code points (in a style similar to Python)

Word boundaries \b, \B

## 9.242.4 Member Function Description

- `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.

- `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`.

- `bool is_valid () const`

Returns whether this object has a valid regular expression assigned.

- `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 `(?:)`).

- `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 `(?:)`).

- `StringArray get_captures () const`

Return a list of all the captures made by the regular expression.

## 9.243 RemoteTransform2D

**Inherits:** `Node2D < CanvasItem < Node < Object`

**Category:** Core

### 9.243.1 Brief Description

### 9.243.2 Member Functions

<code>void</code>	<code>set_remote_node ( NodePath path )</code>
<code>NodePath</code>	<code>get_remote_node () const</code>

### 9.243.3 Member Function Description

- `void set_remote_node ( NodePath path )`
- `NodePath get_remote_node () const`

## 9.244 RenderTargetTexture

**Inherits:** `Texture < Resource < Reference < Object`

**Category:** Core

### 9.244.1 Brief Description

## 9.245 Resource

**Inherits:** `Reference < Object`

**Inherited By:** *Theme, AudioStream, EventStream, CubeMap, Translation, Curve2D, Shape, Shape2D, BakedLight, ColorRamp, StyleBox, Environment, Material, VideoStream, RoomBounds, PackedScene, Texture, Script, Occluder-Polygon2D, Mesh, TileSet, BitMap, Animation, Sample, PolygonPathFinder, Shader, World, SampleLibrary, World2D, Font, SpriteFrames, MeshLibrary, Curve3D, NavigationPolygon, MultiMesh, CanvasItemMaterial, PackedDataContainer, NavigationMesh*

**Category:** Core

### 9.245.1 Brief Description

Base class for all resources.

### 9.245.2 Member Functions

void	<code>set_path ( String path )</code>
void	<code>take_over_path ( String path )</code>
<i>String</i>	<code>get_path () const</code>
void	<code>set_name ( String name )</code>
<i>String</i>	<code>get_name () const</code>
<i>RID</i>	<code>get_rid () const</code>
void	<code>set_import_metadata ( Object metadata )</code>
<i>Object</i>	<code>get_import_metadata () const</code>
<i>Object</i>	<code>duplicate ( bool subresources=false )</code>

### 9.245.3 Signals

- `changed ()`

### 9.245.4 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 instanced 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.

### 9.245.5 Member Function Description

- 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.

- void `take_over_path ( String path )`
- `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.

- 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.

- `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.

- `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 )`
- `Object get_import_metadata () const`
- `Object duplicate ( bool subresources=false )`

## 9.246 ResourcelimportMetadata

**Inherits:** `Reference < Object`

**Category:** Core

### 9.246.1 Brief Description

### 9.246.2 Member Functions

<code>void</code>	<code>set_editor ( String name )</code>
<code>String</code>	<code>get_editor () const</code>
<code>void</code>	<code>add_source ( String path, String md5="" )</code>
<code>String</code>	<code>get_source_path ( int idx ) const</code>
<code>String</code>	<code>get_source_md5 ( int idx ) const</code>
<code>void</code>	<code>remove_source ( int idx )</code>
<code>int</code>	<code>get_source_count () const</code>
<code>void</code>	<code>set_option ( String key, var value )</code>
<code>void</code>	<code>get_option ( String key ) const</code>
<code>StringArray</code>	<code>get_options () const</code>

### 9.246.3 Member Function Description

- `void set_editor ( String name )`
- `String get_editor () const`
- `void add_source ( String path, String md5="" )`
- `String get_source_path ( int idx ) const`
- `String get_source_md5 ( int idx ) const`
- `void remove_source ( int idx )`
- `int get_source_count () const`
- `void set_option ( String key, var value )`
- `void get_option ( String key ) const`
- `StringArray get_options () const`

## 9.247 ResourceInteractiveLoader

Inherits: [Reference](#) < [Object](#)

Category: Core

### 9.247.1 Brief Description

Interactive Resource Loader.

### 9.247.2 Member Functions

<i>Object</i>	<a href="#">get_resource ()</a>
<i>int</i>	<a href="#">poll ()</a>
<i>int</i>	<a href="#">wait ()</a>
<i>int</i>	<a href="#">get_stage () const</a>
<i>int</i>	<a href="#">get_stage_count () const</a>

### 9.247.3 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.

### 9.247.4 Member Function Description

- [Object get\\_resource \(\)](#)

Return the loaded resource (only if loaded). Otherwise, returns null.

- [int poll \(\)](#)

Poll the load. If OK is returned, this means poll will have to be called again. If ERR\_EOF is returned, then the load has finished and the resource can be obtained by calling [get\\_resource](#).

- [int wait \(\)](#)

- [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.

## 9.248 ResourceLoader

Inherits: [Object](#)

Category: Core

### 9.248.1 Brief Description

Resource Loader.

### 9.248.2 Member Functions

<i>ResourceInteractiveLoader</i>	<code>load_interactive ( String path, String type_hint="" )</code>
<i>Resource</i>	<code>load ( String path, String type_hint="", bool p_no_cache=false )</code>
<i>StringArray</i>	<code>get_recognized_extensions_for_type ( String type )</code>
<i>void</i>	<code>set_abort_on_missing_resources ( bool abort )</code>
<i>StringArray</i>	<code>get_dependencies ( String path )</code>
<i>bool</i>	<code>has ( String path )</code>

### 9.248.3 Description

Resource Loader. This is a static object accessible as *ResourceLoader*. GDScript has a simplified load() function, though.

### 9.248.4 Member Function Description

- *ResourceInteractiveLoader* `load_interactive ( String path, String type_hint="" )`

Load a resource interactively, the returned object allows to load with high granularity.

- *Resource* `load ( String path, String type_hint="", bool p_no_cache=false )`
- *StringArray* `get_recognized_extensions_for_type ( String type )`

Return the list of recognized extensions for a resource type.

- *void* `set_abort_on_missing_resources ( bool abort )`

Change the behavior on missing sub-resources. Default is to abort load.

- *StringArray* `get_dependencies ( String path )`
- *bool* `has ( String path )`

## 9.249 ResourcePreloader

**Inherits:** *Node < Object*

**Category:** Core

### 9.249.1 Brief Description

Resource Preloader Node.

## 9.249.2 Member Functions

void	<code>add_resource ( String name, Object resource )</code>
void	<code>remove_resource ( String name )</code>
void	<code>rename_resource ( String name, String newname )</code>
<i>bool</i>	<code>has_resource ( String name ) const</code>
<i>Object</i>	<code>get_resource ( String name ) const</code>
<i>StringArray</i>	<code>get_resource_list ( ) const</code>

## 9.249.3 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.

## 9.249.4 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).

- 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.

- *bool* `has_resource ( String name ) const`

Return true if the preloader has a given resource.

- *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.

## 9.250 ResourceSaver

**Inherits:** *Object*

**Category:** Core

### 9.250.1 Brief Description

Resource Saving Interface.

## 9.250.2 Member Functions

<i>int</i>	<code>save ( String path, Resource resource, int flags=0 )</code>
<i>StringArray</i>	<code>get_recognized_extensions ( Object type )</code>

### 9.250.3 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

### 9.250.4 Description

Resource Saving Interface. This interface is used for saving resources to disk.

### 9.250.5 Member Function Description

- `int save ( String path, Resource resource, int flags=0 )`

Save a resource to disk, to a given path.

- `StringArray get_recognized_extensions ( Object type )`

Return the list of extensions available for saving a resource of a given type.

## 9.251 RichTextLabel

**Inherits:** `Control < CanvasItem < Node < Object`

**Category:** Core

### 9.251.1 Brief Description

Label that displays rich text.

### 9.251.2 Member Functions

<code>void</code>	<code>add_text ( String text )</code>
<code>void</code>	<code>add_image ( Texture image )</code>
<code>void</code>	<code>newline ()</code>
<code>void</code>	<code>push_font ( Object font )</code>
<code>void</code>	<code>push_color ( Color color )</code>
<code>void</code>	<code>push_align ( int align )</code>
<code>void</code>	<code>push_indent ( int level )</code>
<code>void</code>	<code>push_list ( int type )</code>
<code>void</code>	<code>push_meta ( var data )</code>
<code>void</code>	<code>push_underline ()</code>
<code>void</code>	<code>push_table ( int columns )</code>
Continued on next page	

Table 9.22 – continued from previous page

void	<code>set_table_column_expand ( int column, bool expand, int ratio )</code>
void	<code>push_cell ()</code>
void	<code>pop ()</code>
void	<code>clear ()</code>
void	<code>set_meta_underline ( bool enable )</code>
<code>bool</code>	<code>is_meta_underlined () const</code>
void	<code>set_scroll_active ( bool active )</code>
<code>bool</code>	<code>is_scroll_active () const</code>
void	<code>set_scroll_follow ( bool follow )</code>
<code>bool</code>	<code>is_scroll_following () const</code>
<code>Object</code>	<code>get_v_scroll ()</code>
void	<code>scroll_to_line ( int line )</code>
void	<code>set_tab_size ( int spaces )</code>
<code>int</code>	<code>get_tab_size () const</code>
void	<code>set_selection_enabled ( bool enabled )</code>
<code>bool</code>	<code>is_selection_enabled () const</code>
<code>int</code>	<code>parse_bbcode ( String bbcode )</code>
<code>int</code>	<code>append_bbcode ( String bbcode )</code>
void	<code>set_bbcode ( String text )</code>
<code>String</code>	<code>get_bbcode () const</code>
void	<code>set_visible_characters ( int amount )</code>
<code>int</code>	<code>get_visible_characters () const</code>
<code>int</code>	<code>get_total_character_count () const</code>
void	<code>set_use_bbcode ( bool enable )</code>
<code>bool</code>	<code>is_using_bbcode () const</code>

### 9.251.3 Signals

- `meta_clicked ( Nil meta )`

### 9.251.4 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

## 9.251.5 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.

## 9.251.6 Member Function Description

- void **add\_text** ( *String* text )
- void **add\_image** ( *Texture* image )
- void **newline** ( )
- void **push\_font** ( *Object* font )
- void **push\_color** ( *Color* color )
- void **push\_align** ( *int* align )
- void **push\_indent** ( *int* level )
- void **push\_list** ( *int* type )
- void **push\_meta** ( var data )
- void **push\_underline** ( )
- void **push\_table** ( *int* columns )
- void **set\_table\_column\_expand** ( *int* column, *bool* expand, *int* ratio )
- void **push\_cell** ( )
- void **pop** ( )
- void **clear** ( )
- void **set\_meta\_underline** ( *bool* enable )
- *bool* **is\_meta\_underlined** ( ) const
- void **set\_scroll\_active** ( *bool* active )
- *bool* **is\_scroll\_active** ( ) const
- void **set\_scroll\_follow** ( *bool* follow )
- *bool* **is\_scroll\_following** ( ) const
- *Object* **get\_v\_scroll** ( )
- void **scroll\_to\_line** ( *int* line )
- void **set\_tab\_size** ( *int* spaces )

- `int get_tab_size () const`
- `void set_selection_enabled ( bool enabled )`

Set to true if selecting the text inside this richtext is allowed.

- `bool is_selection_enabled () const`

Return true if selecting the text inside this richtext is allowed.

- `int parse_bbcode ( String bbcode )`
- `int append_bbcode ( String bbcode )`
- `void set_bbcode ( String text )`
- `String get_bbcode () const`
- `void set_visible_characters ( int amount )`
- `int get_visible_characters () const`
- `int get_total_character_count () const`
- `void set_use_bbcode ( bool enable )`
- `bool is_using_bbcode () const`

## 9.252 RID

**Category:** Built-In Types

### 9.252.1 Brief Description

### 9.252.2 Member Functions

<code>int</code>	<code>get_id ()</code>
<code>RID</code>	<code>RID ( Object from )</code>

### 9.252.3 Member Function Description

- `int get_id ()`
- `RID RID ( Object from )`

## 9.253 RigidBody

**Inherits:** `PhysicsBody < CollisionObject < Spatial < Node < Object`

**Category:** Core

### 9.253.1 Brief Description

### 9.253.2 Member Functions

void	<code>_integrate_forces ( PhysicsDirectBodyState state ) virtual</code>
void	<code>set_mode ( int mode )</code>
<i>int</i>	<code>get_mode ( ) const</code>
void	<code>set_mass ( float mass )</code>
<i>float</i>	<code>get_mass ( ) const</code>
void	<code>set_weight ( float weight )</code>
<i>float</i>	<code>get_weight ( ) const</code>
void	<code>set_friction ( float friction )</code>
<i>float</i>	<code>get_friction ( ) const</code>
void	<code>set_bounce ( float bounce )</code>
<i>float</i>	<code>get_bounce ( ) const</code>
void	<code>set_linear_velocity ( Vector3 linear_velocity )</code>
<i>Vector3</i>	<code>get_linear_velocity ( ) const</code>
void	<code>set_angular_velocity ( Vector3 angular_velocity )</code>
<i>Vector3</i>	<code>get_angular_velocity ( ) const</code>
void	<code>set_gravity_scale ( float gravity_scale )</code>
<i>float</i>	<code>get_gravity_scale ( ) const</code>
void	<code>set_linear_damp ( float linear_damp )</code>
<i>float</i>	<code>get_linear_damp ( ) const</code>
void	<code>set_angular_damp ( float angular_damp )</code>
<i>float</i>	<code>get_angular_damp ( ) const</code>
void	<code>set_max_contacts_reported ( int amount )</code>
<i>int</i>	<code>get_max_contacts_reported ( ) const</code>
void	<code>set_use_custom_integrator ( bool enable )</code>
<i>bool</i>	<code>is_using_custom_integrator ( )</code>
void	<code>set_contact_monitor ( bool enabled )</code>
<i>bool</i>	<code>is_contact_monitor_enabled ( ) const</code>
void	<code>set_use_continuous_collision_detection ( bool enable )</code>
<i>bool</i>	<code>is_using_continuous_collision_detection ( ) const</code>
void	<code>set_axis_velocity ( Vector3 axis_velocity )</code>
void	<code>apply_impulse ( Vector3 pos, Vector3 impulse )</code>
void	<code>set_sleeping ( bool sleeping )</code>
<i>bool</i>	<code>is_sleeping ( ) const</code>
void	<code>set_can_sleep ( bool able_to_sleep )</code>
<i>bool</i>	<code>is_able_to_sleep ( ) const</code>
void	<code>set_axis_lock ( int axis_lock )</code>
<i>int</i>	<code>get_axis_lock ( ) const</code>
<i>Array</i>	<code>get_colliding_bodies ( ) const</code>

### 9.253.3 Signals

- `body_enter ( Object body )`
- `body_enter_shape ( int body_id, Object body, int body_shape, int local_shape )`
- `body_exit ( Object body )`
- `body_exit_shape ( int body_id, Object body, int body_shape, int local_shape )`

### 9.253.4 Numeric Constants

- `MODE_STATIC = 1`

- **MODE\_KINEMATIC** = 3
- **MODE\_RIGID** = 0
- **MODE\_CHARACTER** = 2

### 9.253.5 Member Function Description

- void **\_integrate\_forces** ( *PhysicsDirectBodyState* state ) virtual
- void **set\_mode** ( *int* mode )
- *int* **get\_mode** ( ) const
- void **set\_mass** ( *float* mass )
- *float* **get\_mass** ( ) const
- void **set\_weight** ( *float* weight )
- *float* **get\_weight** ( ) const
- void **set\_friction** ( *float* friction )
- *float* **get\_friction** ( ) const
- void **set\_bounce** ( *float* bounce )
- *float* **get\_bounce** ( ) const
- void **set\_linear\_velocity** ( *Vector3* linear\_velocity )
- *Vector3* **get\_linear\_velocity** ( ) const
- void **set\_angular\_velocity** ( *Vector3* angular\_velocity )
- *Vector3* **get\_angular\_velocity** ( ) const
- void **set\_gravity\_scale** ( *float* gravity\_scale )
- *float* **get\_gravity\_scale** ( ) const
- void **set\_linear\_damp** ( *float* linear\_damp )
- *float* **get\_linear\_damp** ( ) const
- void **set\_angular\_damp** ( *float* angular\_damp )
- *float* **get\_angular\_damp** ( ) const
- void **set\_max\_contacts\_reported** ( *int* amount )
- *int* **get\_max\_contacts\_reported** ( ) const
- void **set\_use\_custom\_integrator** ( *bool* enable )
- *bool* **is\_using\_custom\_integrator** ( )
- void **set\_contact\_monitor** ( *bool* enabled )
- *bool* **is\_contact\_monitor\_enabled** ( ) const
- void **set\_use\_continuous\_collision\_detection** ( *bool* enable )
- *bool* **is\_using\_continuous\_collision\_detection** ( ) const
- void **set\_axis\_velocity** ( *Vector3* axis\_velocity )
- void **apply\_impulse** ( *Vector3* pos, *Vector3* impulse )

- void `set_sleeping ( bool sleeping )`
- `bool is_sleeping () const`
- void `set_can_sleep ( bool able_to_sleep )`
- `bool is_able_to_sleep () const`
- void `set_axis_lock ( int axis_lock )`
- `int get_axis_lock () const`
- `Array get_colliding_bodies () const`

## 9.254 RigidBody2D

**Inherits:** `PhysicsBody2D < CollisionObject2D < Node2D < CanvasItem < Node < Object`

**Category:** Core

### 9.254.1 Brief Description

Rigid body 2D node.

### 9.254.2 Member Functions

void	<code>_integrate_forces ( Physics2DDirectBodyState state ) virtual</code>
void	<code>set_mode ( int mode )</code>
<code>int</code>	<code>get_mode () const</code>
void	<code>set_mass ( float mass )</code>
<code>float</code>	<code>get_mass () const</code>
void	<code>set_weight ( float weight )</code>
<code>float</code>	<code>get_weight () const</code>
void	<code>set_friction ( float friction )</code>
<code>float</code>	<code>get_friction () const</code>
void	<code>set_bounce ( float bounce )</code>
<code>float</code>	<code>get_bounce () const</code>
void	<code>set_gravity_scale ( float gravity_scale )</code>
<code>float</code>	<code>get_gravity_scale () const</code>
void	<code>set_linear_damp ( float linear_damp )</code>
<code>float</code>	<code>get_linear_damp () const</code>
void	<code>set-angular_damp ( float angular_damp )</code>
<code>float</code>	<code>get-angular_damp () const</code>
void	<code>set_linear_velocity ( Vector2 linear_velocity )</code>
<code>Vector2</code>	<code>get_linear_velocity () const</code>
void	<code>set-angular_velocity ( float angular_velocity )</code>
<code>float</code>	<code>get-angular_velocity () const</code>
void	<code>set_max_contacts_reported ( int amount )</code>
<code>int</code>	<code>get_max_contacts_reported () const</code>
void	<code>set_use_custom_integrator ( bool enable )</code>
<code>bool</code>	<code>is_using_custom_integrator ()</code>

Continued on next page

Table 9.24 – continued from previous page

void	<code>set_contact_monitor ( bool enabled )</code>
<i>bool</i>	<code>is_contact_monitor_enabled () const</code>
void	<code>set_continuous_collision_detection_mode ( int mode )</code>
<i>int</i>	<code>get_continuous_collision_detection_mode () const</code>
void	<code>set_axis_velocity ( Vector2 axis_velocity )</code>
void	<code>apply_impulse ( Vector2 pos, Vector2 impulse )</code>
void	<code>set_applied_force ( Vector2 force )</code>
<i>Vector2</i>	<code>get_applied_force () const</code>
void	<code>set_sleeping ( bool sleeping )</code>
<i>bool</i>	<code>is_sleeping () const</code>
void	<code>set_can_sleep ( bool able_to_sleep )</code>
<i>bool</i>	<code>is_able_to_sleep () const</code>
<i>bool</i>	<code>test_motion ( Vector2 motion, float margin=0.08, Physics2DTestMotionResult result=NULL )</code>
<i>Array</i>	<code>get_colliding_bodies () const</code>

### 9.254.3 Signals

- `body_enter ( Object body )`
- `body_enter_shape ( int body_id, Object body, int body_shape, int local_shape )`
- `body_exit ( Object body )`
- `body_exit_shape ( int body_id, Object body, int body_shape, int local_shape )`

### 9.254.4 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 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.

### 9.254.5 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.

## 9.254.6 Member Function Description

- `void _integrate_forces ( Physics2DDirectBodyState state ) virtual`

Override this function to use a custom force integrator. This allows to hook up to the physics processing and alter the simulation state for the object on every frame.

- `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.

- `int get_mode () const`

Return the current body mode, see `set_mode`.

- `void set_mass ( float mass )`

Set the body mass.

- `float get_mass () const`

Return the body mass.

- `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.

- `float get_weight () const`

Return the body weight given standard earth-weight (gravity 9.8).

- `void set_friction ( float friction )`

Set the body friction, from 0 (frictionless) to 1 (full friction).

- `float get_friction () const`

Return the body friction.

- `void set_bounce ( float bounce )`

Set the body bounciness, from 0 (no bounce) to 1 (full bounce).

- `float get_bounce () const`

Return the body bounciness.

- `void set_gravity_scale ( float gravity_scale )`

Set The gravity factor. This factor multiplies gravity intensity just for this body.

- `float get_gravity_scale () const`

Return the gravity factor.

- `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.

- `float get_linear_damp () const`

Return the linear damp for this body.

- `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.

- `float get_angular_damp () const`

Return the angular damp for this body.

- `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.

- `Vector2 get_linear_velocity () const`

Return the body linear velocity. This changes by physics granularity. See `set_linear_velocity`.

- `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.

- `float get_angular_velocity () const`

Return the body angular velocity. This changes by physics granularity. See `set_angular_velocity`.

- `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.

- `int get_max_contacts_reported () const`

Return the maximum contacts that can be reported. See `set_max_contacts_reported`.

- `void set_use_custom_integrator (bool enable)`

Set to true if the body shall not do any internal force integration at all (like gravity or air friction). Only the `_integrate_forces` will be able to integrate them if overrided.

- `bool is_using_custom_integrator ()`

Return true if the body is not doing any built-in force integration.

- `void set_contact_monitor (bool enabled)`

Enable contact monitoring. This allows the body to emit signals when it collides with another.

- `bool is_contact_monitor_enabled () const`

Return whether contact monitoring is enabled.

- `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.

- `int get_continuous_collision_detection_mode () const`

Return whether this body is using continuous collision detection.

- `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 apply_impulse ( Vector2 pos, 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.

- `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.

- `Vector2 get_applied_force () const`

Return the applied force vector.

- `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.

- `bool is_sleeping () const`

Return whether the body is sleeping.

- `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.

- `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 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).

- `Array get_colliding_bodies () const`

Return a list of the bodies colliding with this one.

## 9.255 Room

**Inherits:** `VisualInstance < Spatial < Node < Object`

**Category:** Core

### 9.255.1 Brief Description

Room data resource.

## 9.255.2 Member Functions

void	<code>set_room ( Room room )</code>
<i>Room</i>	<code>get_room () const</code>
void	<code>compute_room_from_subtree ()</code>
void	<code>set_simulate_acoustics ( bool enable )</code>
<i>bool</i>	<code>is_simulating_acoustics () const</code>

## 9.255.3 Description

Room contains the data to define the bounds of a scene (using a BSP Tree). It is instanced by a RoomInstance node to create rooms. See that class documentation for more information about rooms.

## 9.255.4 Member Function Description

- void `set_room ( Room room )`
- *Room* `get_room () const`
- void `compute_room_from_subtree ()`
- void `set_simulate_acoustics ( bool enable )`
- *bool* `is_simulating_acoustics () const`

## 9.256 RoomBounds

**Inherits:** *Resource* < *Reference* < *Object*

**Category:** Core

### 9.256.1 Brief Description

### 9.256.2 Member Functions

void	<code>set_bounds ( Dictionary bsp_tree )</code>
<i>Dictionary</i>	<code>get_bounds () const</code>
void	<code>set_geometry_hint ( Vector3Array triangles )</code>
<i>Vector3Array</i>	<code>get_geometry_hint () const</code>
void	<code>regenerate_bsp ()</code>
void	<code>regenerate_bsp_cubic ()</code>

### 9.256.3 Member Function Description

- void `set_bounds ( Dictionary bsp_tree )`
- *Dictionary* `get_bounds () const`
- void `set_geometry_hint ( Vector3Array triangles )`
- *Vector3Array* `get_geometry_hint () const`

- void **regenerate\_bsp()**
- void **regenerate\_bsp\_cubic()**

## 9.257 Sample

**Inherits:** *Resource < Reference < Object*

**Category:** Core

### 9.257.1 Brief Description

Audio sample (sound) class.

### 9.257.2 Member Functions

void	<i>create</i> ( <i>int</i> format, <i>bool</i> stereo, <i>int</i> length )
<i>int</i>	<i>get_format</i> () const
<i>bool</i>	<i>is_stereo</i> () const
<i>int</i>	<i>get_length</i> () const
void	<i>set_data</i> ( <i>RawArray</i> data )
<i>RawArray</i>	<i>get_data</i> () const
void	<i>set_mix_rate</i> ( <i>int</i> hz )
<i>int</i>	<i>get_mix_rate</i> () const
void	<i>set_loop_format</i> ( <i>int</i> format )
<i>int</i>	<i>get_loop_format</i> () const
void	<i>set_loop_begin</i> ( <i>int</i> pos )
<i>int</i>	<i>get_loop_begin</i> () const
void	<i>set_loop_end</i> ( <i>int</i> pos )
<i>int</i>	<i>get_loop_end</i> () const

### 9.257.3 Numeric Constants

- **FORMAT\_PCM8 = 0** — 8-bits signed PCM audio.
- **FORMAT\_PCM16 = 1** — 16-bits signed little endian PCM audio.
- **FORMAT\_IMA\_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.

### 9.257.4 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.

## 9.257.5 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.

- `int get_format () const`

Return the sample format.

- `bool is_stereo () const`

Return whether the current sample was created as stereo.

- `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.

- `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.

- `RawArray get_data () const`

Return sample data as little endian.

- `void set_mix_rate ( int hz )`

Set the mix rate for the sample (expected playback frequency).

- `int get_mix_rate () const`

Return the mix rate for the sample.

- `void set_loop_format ( int format )`

Set the loop format (use LOOP\_\* constants as argument).

- `int get_loop_format () const`

Return the loop format.

- `void set_loop_begin ( int pos )`

Set the loop begin position. It must be a valid frame and less than the loop end position.

- `int get_loop_begin () const`

Return the loop begin 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.

- `int get_loop_end () const`

Return the loop end position.

## 9.258 SampleLibrary

**Inherits:** [Resource](#) < [Reference](#) < [Object](#)

**Category:** Core

### 9.258.1 Brief Description

Library that contains a collection of samples.

### 9.258.2 Member Functions

void	<code>add_sample ( String name, Sample sample )</code>
<i>Sample</i>	<code>get_sample ( String name ) const</code>
<i>bool</i>	<code>has_sample ( String name ) const</code>
void	<code>remove_sample ( String name )</code>
void	<code>sample_set_volume_db ( String name, float db )</code>
<i>float</i>	<code>sample_get_volume_db ( String name ) const</code>
void	<code>sample_set_pitch_scale ( String name, float pitch )</code>
<i>float</i>	<code>sample_get_pitch_scale ( String name ) const</code>

### 9.258.3 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.

### 9.258.4 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.

- *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.

- void **sample\_set\_volume\_db** ( *String* name, *float* db )

Set the volume (in dB) 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.

- *float* **sample\_get\_pitch\_scale** ( *String* name ) const

Return the pitch scale for the given sample.

## 9.259 SamplePlayer

Inherits: [Node](#) < [Object](#)

Category: Core

### 9.259.1 Brief Description

Sample Player node.

### 9.259.2 Member Functions

void	<code>set_sample_library ( SampleLibrary library )</code>
<i>SampleLibrary</i>	<code>get_sample_library () const</code>
void	<code>set_polyphony ( int max.voices )</code>
<i>int</i>	<code>get_polyphony () const</code>
<i>int</i>	<code>play ( String name, bool unique=false )</code>
void	<code>stop ( int voice )</code>
void	<code>stop_all ()</code>
void	<code>set_mix_rate ( int voice, int hz )</code>
void	<code>set_pitch_scale ( int voice, float ratio )</code>
void	<code>set_volume ( int voice, float volume )</code>
void	<code>set_volume_db ( int voice, float db )</code>
void	<code>set_pan ( int voice, float pan, float depth=0, float height=0 )</code>
void	<code>set_filter ( int voice, int type, float cutoff_hz, float resonance, float gain=0 )</code>
void	<code>set_chorus ( int voice, float send )</code>
void	<code>set_reverb ( int voice, int room_type, float send )</code>
<i>int</i>	<code>get_mix_rate ( int voice ) const</code>
<i>float</i>	<code>get_pitch_scale ( int voice ) const</code>
<i>float</i>	<code>get_volume ( int voice ) const</code>
<i>float</i>	<code>get_volume_db ( int voice ) const</code>
<i>float</i>	<code>get_pan ( int voice ) const</code>
<i>float</i>	<code>get_pan_depth ( int voice ) const</code>
<i>float</i>	<code>get_pan_height ( int voice ) const</code>
<i>int</i>	<code>get_filter_type ( int voice ) const</code>
<i>float</i>	<code>get_filter_cutoff ( int voice ) const</code>
<i>float</i>	<code>get_filter_resonance ( int voice ) const</code>
<i>float</i>	<code>get_filter_gain ( int voice ) const</code>
<i>float</i>	<code>get_chorus ( int voice ) const</code>
<i>int</i>	<code>get_reverb_room ( int voice ) const</code>
<i>float</i>	<code>get_reverb ( int voice ) const</code>
void	<code>set_default_pitch_scale ( float ratio )</code>
void	<code>set_default_volume ( float volume )</code>
void	<code>set_default_volume_db ( float db )</code>
void	<code>set_default_pan ( float pan, float depth=0, float height=0 )</code>
void	<code>set_default_filter ( int type, float cutoff_hz, float resonance, float gain=0 )</code>

Continued on next page

Table 9.25 – continued from previous page

<code>void</code>	<code>set_default_chorus ( float send )</code>
<code>void</code>	<code>set_default_reverb ( int room_type, float send )</code>
<code>float</code>	<code>get_default_pitch_scale () const</code>
<code>float</code>	<code>get_default_volume () const</code>
<code>float</code>	<code>get_default_volume_db () const</code>
<code>float</code>	<code>get_default_pan () const</code>
<code>float</code>	<code>get_default_pan_depth () const</code>
<code>float</code>	<code>get_default_pan_height () const</code>
<code>int</code>	<code>get_default_filter_type () const</code>
<code>float</code>	<code>get_default_filter_cutoff () const</code>
<code>float</code>	<code>get_default_filter_resonance () const</code>
<code>float</code>	<code>get_default_filter_gain () const</code>
<code>float</code>	<code>get_default_chorus () const</code>
<code>int</code>	<code>get_default_reverb_room () const</code>
<code>float</code>	<code>get_default_reverb () const</code>
<code>bool</code>	<code>is_active () const</code>
<code>bool</code>	<code>is_voice_active ( int voice ) const</code>

### 9.259.3 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.

### 9.259.4 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.

## 9.259.5 Member Function Description

- void **set\_sample\_library** ( *SampleLibrary* library )

Set the sample library for the player.

- *SampleLibrary* **get\_sample\_library** () const

Return the sample library used by the player.

- void **set\_polyphony** ( *int* max\_voices )

Set the polyphony of the player (maximum amount of simultaneous voices).

- *int* **get\_polyphony** () const

Return the polyphony of the player.

- *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 **stop** ( *int* voice )

Stop a given voice.

- void **stop\_all** ()

Stop all playing voices.

- void **set\_mix\_rate** ( *int* voice, *int* hz )

Set the mix rate (in Hz) of a given voice.

- 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\_volume** ( *int* voice, *float* volume )

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 **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\_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_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_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.

- `int get_mix_rate ( int voice ) const`

Return the current mix rate for a given voice.

- `float get_pitch_scale ( int voice ) const`

Return the current pitch scale for a given voice.

- `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.

- `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.

- `int get_filter_type ( int voice ) const`

Return the current filter type in use (see FILTER\_\* constants) for a given voice.

- `float get_filter_cutoff ( int voice ) const`

Return the current filter cutoff frequency for a given voice.

- `float get_filter_resonance ( int voice ) const`

Return the current filter resonance for a given voice.

- `float get_filter_gain ( int voice ) const`

Return the current filter gain for a given voice.

- `float get_chorus ( int voice ) const`

Return the current chorus 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).

- `float get_reverb ( int voice ) const`

Return the current reverberation send level for a given voice.

- `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_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_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_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_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_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.

- `float get_default_pitch_scale () const`

Return the default pitch scale of the player.

- `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_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.

- `int get_default_filter_type () const`

Return the default filter type in use (see FILTER\_\* constants) for the player.

- `float get_default_filter_cutoff () const`

Return the default filter cutoff frequency of the player.

- `float get_default_filter_resonance () const`

Return the default filter resonance of the player.

- `float get_default_filter_gain () const`

Return the default filter gain of the player.

- `float get_default_chorus () const`

Return the default chorus 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_reverb () const`

Return the default reverberation send level of the player.

- `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.

## 9.260 SamplePlayer2D

**Inherits:** `SoundPlayer2D < Node2D < CanvasItem < Node < Object`

**Category:** Core

### 9.260.1 Brief Description

Sample player for positional 2D Sound.

### 9.260.2 Member Functions

<code>void</code>	<code>set_sample_library ( SampleLibrary library )</code>
<code>SampleLibrary</code>	<code>get_sample_library () const</code>
<code>void</code>	<code>set_polyphony ( int max_voices )</code>
<code>int</code>	<code>get_polyphony () const</code>
<code>int</code>	<code>play ( String sample, int voice=-2 )</code>
<code>void</code>	<code>voice_set_pitch_scale ( int voice, float ratio )</code>
<code>void</code>	<code>voice_set_volume_scale_db ( int voice, float db )</code>
<code>bool</code>	<code>is_voice_active ( int voice ) const</code>
<code>void</code>	<code>stop_voice ( int voice )</code>
<code>void</code>	<code>stop_all ()</code>
<code>void</code>	<code>set_random_pitch_scale ( float val )</code>
<code>float</code>	<code>get_random_pitch_scale () const</code>

### 9.260.3 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.

## 9.260.4 Description

Sample player for positional 2D Sound. Plays sound samples positionally, left and right depending on the distance/place on the screen.

## 9.260.5 Member Function Description

- `void set_sample_library ( SampleLibrary library )`

Set the sample library for the player.

- `SampleLibrary get_sample_library ( ) const`

Return the sample library used by the player.

- `void set_polyphony ( int max.voices )`

Set the polyphony of the player (maximum amount of simultaneous voices).

- `int get_polyphony ( ) const`

Return the polyphony of the player.

- `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 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.

- `bool is_voice_active ( int voice ) const`

Return whether a voice is still active or has stopped playing.

- `void stop_voice ( int voice )`

Stop a given voice.

- `void stop_all ( )`

Stop all playing 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:  $1.0 / (1.0 - \text{variation})$

- `float get_random_pitch_scale ( ) const`

Return the amplitude used for random pitch scale variations.

## 9.261 SceneState

**Inherits:** [Reference](#) < [Object](#)

**Category:** Core

### 9.261.1 Brief Description

### 9.261.2 Member Functions

<i>int</i>	<code>get_node_count () const</code>
<i>String</i>	<code>get_node_type ( int idx ) const</code>
<i>String</i>	<code>get_node_name ( int idx ) const</code>
<i>NodePath</i>	<code>get_node_path ( int idx, bool for_parent=false ) const</code>
<i>NodePath</i>	<code>get_node_owner_path ( int idx ) const</code>
<i>PackedScene</i>	<code>get_node_instance ( int idx ) const</code>
<i>StringArray</i>	<code>get_node_groups ( int idx ) const</code>
<i>int</i>	<code>get_node_property_count ( int idx ) const</code>
<i>String</i>	<code>get_node_property_name ( int idx, int prop_idx ) const</code>
<i>void</i>	<code>get_node_property_value ( int idx, int prop_idx ) const</code>
<i>int</i>	<code>get_connection_count () const</code>
<i>NodePath</i>	<code>get_connection_source ( int idx ) const</code>
<i>String</i>	<code>get_connection_signal ( int idx ) const</code>
<i>NodePath</i>	<code>get_connection_target ( int idx ) const</code>
<i>String</i>	<code>get_connection_method ( int idx ) const</code>
<i>int</i>	<code>get_connection_flags ( int idx ) const</code>
<i>Array</i>	<code>get_connection_binds ( int idx ) const</code>

### 9.261.3 Member Function Description

- `int get_node_count () const`
- `String get_node_type ( int idx ) const`
- `String get_node_name ( int idx ) const`
- `NodePath get_node_path ( int idx, bool for_parent=false ) const`
- `NodePath get_node_owner_path ( int idx ) const`
- `PackedScene get_node_instance ( int idx ) const`
- `StringArray get_node_groups ( int idx ) 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`
- `int get_connection_count () const`
- `NodePath get_connection_source ( int idx ) const`
- `String get_connection_signal ( int idx ) const`
- `NodePath get_connection_target ( int idx ) const`

- `String get_connection_method ( int idx ) const`
- `int get_connection_flags ( int idx ) const`
- `Array get_connection_binds ( int idx ) const`

## 9.262 SceneTree

Inherits: `MainLoop < Object`

Category: Core

### 9.262.1 Brief Description

### 9.262.2 Member Functions

<code>void</code>	<code>notify_group ( int call_flags, String group, int notification )</code>
<code>void</code>	<code>set_group ( int call_flags, String group, String property, var value )</code>
<code>Array</code>	<code>get_nodes_in_group ( String group )</code>
<code>View-port</code>	<code>get_root () const</code>
<code>bool</code>	<code>has_group ( String name ) const</code>
<code>void</code>	<code>set_auto_accept_quit ( bool enabled )</code>
<code>void</code>	<code>set_editor_hint ( bool enable )</code>
<code>bool</code>	<code>is_editor_hint () const</code>
<code>void</code>	<code>set_debug_collisions_hint ( bool enable )</code>
<code>bool</code>	<code>is_debugging_collisions_hint () const</code>
<code>void</code>	<code>set_debug_navigation_hint ( bool enable )</code>
<code>bool</code>	<code>is_debugging_navigation_hint () const</code>
<code>void</code>	<code>set_edited_scene_root ( Object scene )</code>
<code>Object</code>	<code>get_edited_scene_root () const</code>
<code>void</code>	<code>set_pause ( bool enable )</code>
<code>bool</code>	<code>is_paused () const</code>
<code>void</code>	<code>set_input_as_handled ()</code>
<code>int</code>	<code>get_node_count () const</code>
<code>int</code>	<code>get_frame () const</code>
<code>void</code>	<code>quit ()</code>
<code>void</code>	<code>set_screen_stretch ( int mode, int aspect, Vector2 minsize )</code>
<code>void</code>	<code>queue_delete ( Object obj )</code>
<code>void</code>	<code>call_group ( int flags, String group, String method, var arg0=NULL, var arg1=NULL, var arg2=NULL, var arg3=NULL, var arg4=NULL )</code>
<code>void</code>	<code>set_current_scene ( Node child_node )</code>
<code>Node</code>	<code>get_current_scene () const</code>
<code>int</code>	<code>change_scene ( String path )</code>
<code>int</code>	<code>change_scene_to ( PackedScene packed_scene )</code>
<code>int</code>	<code>reload_current_scene ()</code>

### 9.262.3 Signals

- `screen_resized ()`

- **node\_removed** ( *Object* node )
- **idle\_frame** ( )
- **tree\_changed** ( )
- **fixed\_frame** ( )

#### 9.262.4 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

#### 9.262.5 Member Function Description

- void **notify\_group** ( *int* call\_flags, *String* group, *int* notification )
- void **set\_group** ( *int* call\_flags, *String* group, *String* property, var value )
- *Array* **get\_nodes\_in\_group** ( *String* group )
- *Viewport* **get\_root** ( ) const
- *bool* **has\_group** ( *String* name ) const
- void **set\_auto\_accept\_quit** ( *bool* enabled )
- void **set\_editor\_hint** ( *bool* enable )
- *bool* **is\_editor\_hint** ( ) const
- void **set\_debug\_collisions\_hint** ( *bool* enable )
- *bool* **is\_debugging\_collisions\_hint** ( ) const
- void **set\_debug\_navigation\_hint** ( *bool* enable )
- *bool* **is\_debugging\_navigation\_hint** ( ) const
- void **set\_edited\_scene\_root** ( *Object* scene )
- *Object* **get\_edited\_scene\_root** ( ) const
- void **set\_pause** ( *bool* enable )
- *bool* **is\_paused** ( ) const
- void **set\_input\_as\_handled** ( )

- `int get_node_count () const`
- `int get_frame () const`
- `void quit ()`
- `void set_screen_stretch ( int mode, int aspect, Vector2 minsize )`
- `void queue_delete ( Object obj )`
- `void call_group ( int flags, String group, String method, var arg0=NULL, var arg1=NULL, var arg2=NULL, var arg3=NULL, var arg4=NULL )`
- `void set_current_scene ( Node child_node )`
- `Node get_current_scene () const`
- `int change_scene ( String path )`
- `int change_scene_to ( PackedScene packed_scene )`
- `int reload_current_scene ()`

## 9.263 Script

**Inherits:** `Resource < Reference < Object`

**Inherited By:** `GDScript`

**Category:** Core

### 9.263.1 Brief Description

Base class for scripts.

### 9.263.2 Member Functions

<code>bool</code>	<code>can_instance () const</code>
<code>bool</code>	<code>instance_has ( Object base_object ) const</code>
<code>bool</code>	<code>has_source_code () const</code>
<code>String</code>	<code>get_source_code () const</code>
<code>void</code>	<code>set_source_code ( String source )</code>
<code>int</code>	<code>reload ()</code>

### 9.263.3 Description

Base class for scripts. Any script that is loaded becomes one of these resources, which can then create instances.

### 9.263.4 Member Function Description

- `bool can_instance () const`

Return true if this script can be instance (ie not a library).

- `bool instance_has ( Object base_object ) const`

Return true if a given object uses an instance of this script.

- `bool has_source_code () const`

Return true if the script contains source code.

- `String get_source_code () const`

Return the script source code (if available).

- `void set_source_code ( String source )`

Set the script source code.

- `int reload ()`

Reload the script. This will fail if there are existing instances.

## 9.264 ScrollBar

**Inherits:** `Range < Control < CanvasItem < Node < Object`

**Inherited By:** `HScrollBar, VScrollBar`

**Category:** Core

### 9.264.1 Brief Description

Base class for scroll bars.

### 9.264.2 Member Functions

<code>void</code>	<code>set_custom_step ( float step )</code>
<code>float</code>	<code>get_custom_step () const</code>

### 9.264.3 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.

### 9.264.4 Member Function Description

- `void set_custom_step ( float step )`
- `float get_custom_step () const`

## 9.265 ScrollContainer

**Inherits:** `Container < Control < CanvasItem < Node < Object`

**Category:** Core

### 9.265.1 Brief Description

A helper node for displaying scrollable elements (e.g. lists).

### 9.265.2 Member Functions

void	<code>set_enable_h_scroll ( bool enable )</code>
<i>bool</i>	<code>is_h_scroll_enabled () const</code>
void	<code>set_enable_v_scroll ( bool enable )</code>
<i>bool</i>	<code>is_v_scroll_enabled () const</code>
void	<code>set_h_scroll ( int val )</code>
<i>int</i>	<code>get_h_scroll () const</code>
void	<code>set_v_scroll ( int val )</code>
<i>int</i>	<code>get_v_scroll () const</code>

### 9.265.3 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.

### 9.265.4 Member Function Description

- void `set_enable_h_scroll ( bool enable )`
- *bool* `is_h_scroll_enabled () const`
- void `set_enable_v_scroll ( bool enable )`
- *bool* `is_v_scroll_enabled () const`
- void `set_h_scroll ( int val )`
- *int* `get_h_scroll () const`
- void `set_v_scroll ( int val )`
- *int* `get_v_scroll () const`

## 9.266 SegmentShape2D

**Inherits:** [Shape2D](#) < [Resource](#) < [Reference](#) < [Object](#)

**Category:** Core

### 9.266.1 Brief Description

Segment Shape for 2D Collision Detection.

## 9.266.2 Member Functions

void	<code>set_a ( Vector2 a )</code>
<code>Vector2</code>	<code>get_a () const</code>
void	<code>set_b ( Vector2 b )</code>
<code>Vector2</code>	<code>get_b () const</code>

## 9.266.3 Description

Segment Shape for 2D Collision Detection, consists of two points, ‘a’ and ‘b’.

## 9.266.4 Member Function Description

- void `set_a ( Vector2 a )`

Set the first point’s position.

- `Vector2 get_a () const`

Return the first point’s position.

- void `set_b ( Vector2 b )`

Set the second point’s position.

- `Vector2 get_b () const`

Return the second point’s position.

## 9.267 Semaphore

**Inherits:** [Reference < Object](#)

**Category:** Core

## 9.267.1 Brief Description

## 9.267.2 Member Functions

Error	<code>wait ()</code>
Error	<code>post ()</code>

## 9.267.3 Member Function Description

- Error `wait ()`
- Error `post ()`

## 9.268 Separator

**Inherits:** [Control](#) < [CanvasItem](#) < [Node](#) < [Object](#)

**Inherited By:** [VSeparator](#), [HSeparator](#)

**Category:** Core

### 9.268.1 Brief Description

Base class for separators.

### 9.268.2 Description

Separator is a [Control](#) used for separating other controls. It's purely a visual decoration. Horizontal ([HSeparator](#)) and Vertical ([VSeparator](#)) versions are available.

## 9.269 Shader

**Inherits:** [Resource](#) < [Reference](#) < [Object](#)

**Inherited By:** [MaterialShader](#), [CanvasItemShader](#), [ShaderGraph](#)

**Category:** Core

### 9.269.1 Brief Description

To be changed, ignore.

### 9.269.2 Member Functions

<code>int</code>	<code>get_mode () const</code>
<code>void</code>	<code>set_code ( String vcode, String fcode, String lcode, int fofs=0, int lofs=0 )</code>
<code>String</code>	<code>get_vertex_code () const</code>
<code>String</code>	<code>get_fragment_code () const</code>
<code>String</code>	<code>get_light_code () const</code>
<code>void</code>	<code>set_default_texture_param ( String param, Texture texture )</code>
<code>Texture</code>	<code>get_default_texture_param ( String param ) const</code>
<code>bool</code>	<code>has_param ( String name ) const</code>

### 9.269.3 Numeric Constants

- **MODE\_MATERIAL = 0**
- **MODE\_CANVAS\_ITEM = 1**
- **MODE\_POST\_PROCESS = 2**

## 9.269.4 Description

To be changed, ignore.

## 9.269.5 Member Function Description

- `int get_mode () const`
- `void set_code ( String vcode, String fcode, String lcode, int fofofs=0, int lofs=0 )`
- `String get_vertex_code () const`
- `String get_fragment_code () const`
- `String get_light_code () const`
- `void set_default_texture_param ( String param, Texture texture )`
- `Texture get_default_texture_param ( String param ) const`
- `bool has_param ( String name ) const`

## 9.270 ShaderGraph

**Inherits:** `Shader < Resource < Reference < Object`

**Inherited By:** `MaterialShaderGraph, CanvasItemShaderGraph`

**Category:** Core

### 9.270.1 Brief Description

### 9.270.2 Member Functions

<code>void</code>	<code>node_add ( int shader_type, int node_type, int id )</code>
<code>void</code>	<code>node_remove ( int shader_type, int id )</code>
<code>void</code>	<code>node_set_pos ( int shader_type, int id, Vector2 pos )</code>
<code>Vector2</code>	<code>node_get_pos ( int shader_type, int id ) const</code>
<code>int</code>	<code>node_get_type ( int shader_type, int id ) const</code>
<code>Array</code>	<code>get_node_list ( int shader_type ) const</code>
<code>void</code>	<code>default_set_value ( int shader_type, int id, int param_id, var value )</code>
<code>void</code>	<code>default_get_value ( int shader_type, int id, int param_id )</code>
<code>void</code>	<code>scalar_const_node_set_value ( int shader_type, int id, float value )</code>
<code>float</code>	<code>scalar_const_node_get_value ( int shader_type, int id ) const</code>
<code>void</code>	<code>vec_const_node_set_value ( int shader_type, int id, Vector3 value )</code>
<code>Vector3</code>	<code>vec_const_node_get_value ( int shader_type, int id ) const</code>
<code>void</code>	<code>rgb_const_node_set_value ( int shader_type, int id, Color value )</code>
<code>Color</code>	<code>rgb_const_node_get_value ( int shader_type, int id ) const</code>
<code>void</code>	<code>xform_const_node_set_value ( int shader_type, int id, Transform value )</code>
<code>Transform</code>	<code>xform_const_node_get_value ( int shader_type, int id ) const</code>
<code>void</code>	<code>texture_node_set_filter_size ( int shader_type, int id, int filter_size )</code>
<code>int</code>	<code>texture_node_get_filter_size ( int shader_type, int id ) const</code>

Continued on next page

Table 9.26 – continued from previous page

void	<i>texture_node_set_filter_strength</i> ( int shader_type, float id, float filter_strength )
float	<i>texture_node_get_filter_strength</i> ( int shader_type, float id ) const
void	<i>scalar_op_node_set_op</i> ( int shader_type, float id, int op )
int	<i>scalar_op_node_get_op</i> ( int shader_type, float id ) const
void	<i>vec_op_node_set_op</i> ( int shader_type, float id, int op )
int	<i>vec_op_node_get_op</i> ( int shader_type, float id ) const
void	<i>vec_scalar_op_node_set_op</i> ( int shader_type, float id, int op )
int	<i>vec_scalar_op_node_get_op</i> ( int shader_type, float id ) const
void	<i>rgb_op_node_set_op</i> ( int shader_type, float id, int op )
int	<i>rgb_op_node_get_op</i> ( int shader_type, float id ) const
void	<i>xform_vec_mult_node_set_no_translation</i> ( int shader_type, int id, bool disable )
bool	<i>xform_vec_mult_node_get_no_translation</i> ( int shader_type, int id ) const
void	<i>scalar_func_node_set_function</i> ( int shader_type, int id, int func )
int	<i>scalar_func_node_get_function</i> ( int shader_type, int id ) const
void	<i>vec_func_node_set_function</i> ( int shader_type, int id, int func )
int	<i>vec_func_node_get_function</i> ( int shader_type, int id ) const
void	<i>input_node_set_name</i> ( int shader_type, int id, String name )
String	<i>input_node_get_name</i> ( int shader_type, int id )
void	<i>scalar_input_node_set_value</i> ( int shader_type, int id, float value )
float	<i>scalar_input_node_get_value</i> ( int shader_type, int id ) const
void	<i>vec_input_node_set_value</i> ( int shader_type, int id, Vector3 value )
Vector3	<i>vec_input_node_get_value</i> ( int shader_type, int id ) const
void	<i>rgb_input_node_set_value</i> ( int shader_type, int id, Color value )
Color	<i>rgb_input_node_get_value</i> ( int shader_type, int id ) const
void	<i>xform_input_node_set_value</i> ( int shader_type, int id, Transform value )
Transform	<i>xform_input_node_get_value</i> ( int shader_type, int id ) const
void	<i>texture_input_node_set_value</i> ( int shader_type, int id, Texture value )
Texture	<i>texture_input_node_get_value</i> ( int shader_type, int id ) const
void	<i>cubemap_input_node_set_value</i> ( int shader_type, int id, CubeMap value )
CubeMap	<i>cubemap_input_node_get_value</i> ( int shader_type, int id ) const
void	<i>comment_node_set_text</i> ( int shader_type, int id, String text )
String	<i>comment_node_get_text</i> ( int shader_type, int id ) const
void	<i>color_ramp_node_set_ramp</i> ( int shader_type, int id, ColorArray colors, RealArray offsets )
ColorArray	<i>color_ramp_node_get_colors</i> ( int shader_type, int id ) const
RealArray	<i>color_ramp_node_get_offsets</i> ( int shader_type, int id ) const
void	<i>curve_map_node_set_points</i> ( int shader_type, int id, Vector2Array points )
Vector2Array	<i>curve_map_node_get_points</i> ( int shader_type, int id ) const
Error	<i>connect_node</i> ( int shader_type, int src_id, int src_slot, int dst_id, int dst_slot )
bool	<i>is_node_connected</i> ( int shader_type, int src_id, int src_slot, int dst_id, int dst_slot ) const
void	<i>disconnect_node</i> ( int shader_type, int src_id, int src_slot, int dst_id, int dst_slot )
Array	<i>get_node_connections</i> ( int shader_type ) const
void	<i>clear</i> ( int shader_type )
void	<i>node_set_state</i> ( int shader_type, int id, var state )
Variant	<i>node_get_state</i> ( int shader_type, int id ) const

### 9.270.3 Signals

- **updated ()**

## 9.270.4 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

## 9.270.5 Member Function Description

- void **node\_add** ( *int* shader\_type, *int* node\_type, *int* id )
- void **node\_remove** ( *int* shader\_type, *int* id )
- void **node\_set\_pos** ( *int* shader\_type, *int* id, *Vector2* pos )
- *Vector2* **node\_get\_pos** ( *int* shader\_type, *int* id ) const
- *int* **node\_get\_type** ( *int* shader\_type, *int* id ) const
- *Array* **get\_node\_list** ( *int* shader\_type ) const
- void **default\_set\_value** ( *int* shader\_type, *int* id, *int* param\_id, var value )
- void **default\_get\_value** ( *int* shader\_type, *int* id, *int* param\_id )
- void **scalar\_const\_node\_set\_value** ( *int* shader\_type, *int* id, *float* value )
- *float* **scalar\_const\_node\_get\_value** ( *int* shader\_type, *int* id ) const
- void **vec\_const\_node\_set\_value** ( *int* shader\_type, *int* id, *Vector3* value )
- *Vector3* **vec\_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\_const\_node\_get\_value** ( *int* shader\_type, *int* id ) const
- void **xform\_const\_node\_set\_value** ( *int* shader\_type, *int* id, *Transform* value )
- *Transform* **xform\_const\_node\_get\_value** ( *int* shader\_type, *int* id ) const
- void **texture\_node\_set\_filter\_size** ( *int* shader\_type, *int* id, *int* filter\_size )
- *int* **texture\_node\_get\_filter\_size** ( *int* shader\_type, *int* id ) const
- void **texture\_node\_set\_filter\_strength** ( *int* shader\_type, *float* id, *float* filter\_strength )
- *float* **texture\_node\_get\_filter\_strength** ( *int* shader\_type, *float* id ) const
- void **scalar\_op\_node\_set\_op** ( *int* shader\_type, *float* id, *int* op )
- *int* **scalar\_op\_node\_get\_op** ( *int* shader\_type, *float* id ) const
- void **vec\_op\_node\_set\_op** ( *int* shader\_type, *float* id, *int* op )
- *int* **vec\_op\_node\_get\_op** ( *int* shader\_type, *float* id ) const
- void **vec\_scalar\_op\_node\_set\_op** ( *int* shader\_type, *float* id, *int* op )
- *int* **vec\_scalar\_op\_node\_get\_op** ( *int* shader\_type, *float* id ) const
- void **rgb\_op\_node\_set\_op** ( *int* shader\_type, *float* id, *int* op )
- *int* **rgb\_op\_node\_get\_op** ( *int* shader\_type, *float* id ) const
- void **xform\_vec\_mult\_node\_set\_no\_translation** ( *int* shader\_type, *int* id, *bool* disable )

- `bool xform_vec_mult_node_get_no_translation ( int shader_type, int id ) const`
- `void scalar_func_node_set_function ( int shader_type, int id, int func )`
- `int scalar_func_node_get_function ( int shader_type, int id ) const`
- `void vec_func_node_set_function ( int shader_type, int id, int func )`
- `int vec_func_node_get_function ( int shader_type, int id ) const`
- `void input_node_set_name ( int shader_type, int id, String name )`
- `String input_node_get_name ( int shader_type, int id )`
- `void scalar_input_node_set_value ( int shader_type, int id, float value )`
- `float scalar_input_node_get_value ( int shader_type, int id ) const`
- `void vec_input_node_set_value ( int shader_type, int id, Vector3 value )`
- `Vector3 vec_input_node_get_value ( int shader_type, int id ) const`
- `void rgb_input_node_set_value ( int shader_type, int id, Color value )`
- `Color rgb_input_node_get_value ( int shader_type, int id ) const`
- `void xform_input_node_set_value ( int shader_type, int id, Transform value )`
- `Transform xform_input_node_get_value ( int shader_type, int id ) const`
- `void texture_input_node_set_value ( int shader_type, int id, Texture value )`
- `Texture texture_input_node_get_value ( int shader_type, int id ) const`
- `void cubemap_input_node_set_value ( int shader_type, int id, CubeMap value )`
- `CubeMap cubemap_input_node_get_value ( int shader_type, int id ) const`
- `void comment_node_set_text ( int shader_type, int id, String text )`
- `String comment_node_get_text ( int shader_type, int id ) const`
- `void color_ramp_node_set_ramp ( int shader_type, int id, ColorArray colors, RealArray offsets )`
- `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 curve_map_node_set_points ( int shader_type, int id, Vector2Array points )`
- `Vector2Array curve_map_node_get_points ( int shader_type, int id ) const`
- `Error connect_node ( int shader_type, int src_id, int src_slot, int dst_id, int dst_slot )`
- `bool is_node_connected ( int shader_type, int src_id, int src_slot, int dst_id, int dst_slot ) const`
- `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`
- `void clear ( int shader_type )`
- `void node_set_state ( int shader_type, int id, var state )`
- `Variant node_get_state ( int shader_type, int id ) const`

## 9.271 ShaderMaterial

**Inherits:** [Material](#) < [Resource](#) < [Reference](#) < [Object](#)

**Category:** Core

### 9.271.1 Brief Description

### 9.271.2 Member Functions

void	<code>set_shader ( <a href="#">Shader</a> shader )</code>
<a href="#">Shader</a>	<code>get_shader ( ) const</code>
void	<code>set_shader_param ( <a href="#">String</a> param, Variant value )</code>
Variant	<code>get_shader_param ( <a href="#">String</a> param ) const</code>

### 9.271.3 Member Function Description

- void `set_shader ( Shader shader )`
- [Shader](#) `get_shader ( ) const`
- void `set_shader_param ( String param, Variant value )`
- Variant `get_shader_param ( String param ) const`

## 9.272 Shape

**Inherits:** [Resource](#) < [Reference](#) < [Object](#)

**Inherited By:** [SphereShape](#), [PlaneShape](#), [CapsuleShape](#), [BoxShape](#), [ConvexPolygonShape](#), [RayShape](#), [ConcavePolygonShape](#)

**Category:** Core

### 9.272.1 Brief Description

## 9.273 Shape2D

**Inherits:** [Resource](#) < [Reference](#) < [Object](#)

**Inherited By:** [RayShape2D](#), [CapsuleShape2D](#), [LineShape2D](#), [CircleShape2D](#), [ConcavePolygonShape2D](#), [ConvexPolygonShape2D](#), [RectangleShape2D](#), [SegmentShape2D](#)

**Category:** Core

### 9.273.1 Brief Description

Base class for all 2D Shapes.

## 9.273.2 Member Functions

void	<code>set_custom_solver_bias (float bias)</code>
float	<code>get_custom_solver_bias () const</code>
bool	<code>collide (Matrix32 local_xform, Shape2D with_shape, Matrix32 shape_xform)</code>
bool	<code>collide_with_motion (Matrix32 local_xform, Vector2 local_motion, Shape2D with_shape, Matrix32 shape_xform, Vector2 shape_motion)</code>
Variant	<code>collide_and_get_contacts (Matrix32 local_xform, Shape2D with_shape, Matrix32 shape_xform)</code>
Variant	<code>collide_with_motion_and_get_contacts (Matrix32 local_xform, Vector2 local_motion, Shape2D with_shape, Matrix32 shape_xform, Vector2 shape_motion)</code>

## 9.273.3 Description

Base class for all 2D Shapes. All 2D shape types inherit from this.

## 9.273.4 Member Function Description

- 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.

- `float get_custom_solver_bias () const`

Return the custom solver bias.

- `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`).

- `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_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`).

- 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`).

## 9.274 Skeleton

**Inherits:** `Spatial < Node < Object`

**Category:** Core

### 9.274.1 Brief Description

Skeleton for characters and animated objects.

### 9.274.2 Member Functions

<code>void</code>	<code>add_bone ( String name )</code>
<code>int</code>	<code>find_bone ( String name ) const</code>
<code>String</code>	<code>get_bone_name ( int bone_idx ) const</code>
<code>int</code>	<code>get_bone_parent ( int bone_idx ) const</code>
<code>void</code>	<code>set_bone_parent ( int bone_idx, int parent_idx )</code>
<code>int</code>	<code>get_bone_count ( ) const</code>
<code>void</code>	<code>unparent_bone_and_rest ( int bone_idx )</code>
<code>Transform</code>	<code>get_bone_rest ( int bone_idx ) const</code>
<code>void</code>	<code>set_bone_rest ( int bone_idx, Transform rest )</code>
<code>void</code>	<code>set_bone_disable_rest ( int bone_idx, bool disable )</code>
<code>bool</code>	<code>is_bone_rest_disabled ( int bone_idx ) const</code>
<code>void</code>	<code>bind_child_node_to_bone ( int bone_idx, Node node )</code>
<code>void</code>	<code>unbind_child_node_from_bone ( int bone_idx, Node node )</code>
<code>Array</code>	<code>get_bound_child_nodes_to_bone ( int bone_idx ) const</code>
<code>void</code>	<code>clear_bones ( )</code>
<code>Transform</code>	<code>get_bone_pose ( int bone_idx ) const</code>
<code>void</code>	<code>set_bone_pose ( int bone_idx, Transform pose )</code>
<code>void</code>	<code>set_bone_global_pose ( int bone_idx, Transform pose )</code>
<code>Transform</code>	<code>get_bone_global_pose ( int bone_idx ) const</code>
<code>Transform</code>	<code>get_bone_custom_pose ( int bone_idx ) const</code>
<code>void</code>	<code>set_bone_custom_pose ( int bone_idx, Transform custom_pose )</code>
<code>Transform</code>	<code>get_bone_transform ( int bone_idx ) const</code>

### 9.274.3 Numeric Constants

- `NOTIFICATION_UPDATE_SKELETON = 50`

### 9.274.4 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.

## 9.274.5 Member Function Description

- `void add_bone ( String name )`

Add a bone, with name “name”. `get_bone_count` will become the bone index.

- `int find_bone ( String name ) const`

Return the bone index that matches “name” as its name.

- `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”.

- `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”.

- `int get_bone_count ( ) const`

Return the amount of bones in the skeleton.

- `void unparent_bone_and_rest ( int bone_idx )`

- `Transform get_bone_rest ( int bone_idx ) const`

Return the rest transform for a bone “bone\_idx”.

- `void set_bone_rest ( int bone_idx, Transform rest )`

Set the rest transform for bone “bone\_idx”

- `void set_bone_disable_rest ( int bone_idx, bool disable )`

- `bool is_bone_rest_disabled ( int bone_idx ) const`

- `void bind_child_node_to_bone ( int bone_idx, Node node )`

Deprecated soon.

- `void unbind_child_node_from_bone ( int bone_idx, Node node )`

Deprecated soon.

- `Array get_bound_child_nodes_to_bone ( int bone_idx ) const`

Deprecated soon.

- `void clear_bones ( )`

Clear all the bones in this skeleton.

- `Transform get_bone_pose ( int bone_idx ) const`

Return the pose transform for bone “bone\_idx”.

- `void set_bone_pose ( int bone_idx, Transform pose )`

Return the pose transform for bone “bone\_idx”.

- `void set_bone_global_pose ( int bone_idx, Transform pose )`

- `Transform get_bone_global_pose ( int bone_idx ) const`

- `Transform get_bone_custom_pose ( int bone_idx ) const`
- `void set_bone_custom_pose ( int bone_idx, Transform custom_pose )`
- `Transform get_bone_transform ( int bone_idx ) const`

## 9.275 Slider

**Inherits:** `Range < Control < CanvasItem < Node < Object`

**Inherited By:** `HSlider, VSlider`

**Category:** Core

### 9.275.1 Brief Description

Base class for GUI Sliders.

### 9.275.2 Member Functions

void	<code>set_ticks ( int count )</code>
<code>int</code>	<code>get_ticks () const</code>
<code>bool</code>	<code>get_ticks_on_borders () const</code>
void	<code>set_ticks_on_borders ( bool ticks_on_border )</code>

### 9.275.3 Description

Base class for GUI Sliders.

### 9.275.4 Member Function Description

- `void set_ticks ( int count )`

Set amount of ticks to display in slider.

- `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_on_borders ( bool ticks_on_border )`

Set true if ticks are visible on borders.

## 9.276 SliderJoint

**Inherits:** `Joint < Spatial < Node < Object`

**Category:** Core

### 9.276.1 Brief Description

### 9.276.2 Member Functions

void	<code>set_param ( int param, float value )</code>
<code>float</code>	<code>get_param ( int param ) const</code>

### 9.276.3 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`

### 9.276.4 Member Function Description

- void `set_param ( int param, float value )`
- `float get_param ( int param ) const`

## 9.277 SoundPlayer2D

**Inherits:** [Node2D](#) < [CanvasItem](#) < [Node](#) < [Object](#)

**Inherited By:** [SamplePlayer2D](#)

**Category:** Core

### 9.277.1 Brief Description

Base class for playing spatial 2D sound.

### 9.277.2 Member Functions

void	<code>set_param ( int param, float value )</code>
float	<code>get_param ( int param ) const</code>

### 9.277.3 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`

### 9.277.4 Description

Base class for playing spatial 2D sound.

### 9.277.5 Member Function Description

- void `set_param ( int param, float value )`
- `float get_param ( int param ) const`

## 9.278 SoundRoomParams

**Inherits:** [Node](#) < [Object](#)

**Category:** Core

## 9.278.1 Brief Description

## 9.278.2 Member Functions

void	<code>set_param ( int param, float value )</code>
<code>float</code>	<code>get_param ( int param ) const</code>
void	<code>set_reverb_mode ( int reverb_mode )</code>
<code>int</code>	<code>get_reverb_mode () const</code>
void	<code>set_force_params_to_all_sources ( bool enabled )</code>
<code>bool</code>	<code>is_forcing_params_to_all_sources ()</code>

## 9.278.3 Member Function Description

- void `set_param ( int param, float value )`
- `float get_param ( int param ) const`
- void `set_reverb_mode ( int reverb_mode )`
- `int get_reverb_mode () const`
- void `set_force_params_to_all_sources ( bool enabled )`
- `bool is_forcing_params_to_all_sources ()`

## 9.279 Spatial

**Inherits:** `Node < Object`

**Inherited By:** `Joint, RayCast, Camera, BoneAttachment, CollisionShape, Path, VisualInstance, VehicleWheel, Position3D, ProximityGroup, SpatialPlayer, WorldEnvironment, PathFollow, NavigationMeshInstance, VisibilityNotifier, Navigation, CollisionPolygon, GridMap, Skeleton, CollisionObject`

**Category:** Core

## 9.279.1 Brief Description

Base class for all 3D nodes.

## 9.279.2 Member Functions

void	<code>set_transform ( Transform local )</code>
<code>Transform</code>	<code>get_transform () const</code>
void	<code>set_translation ( Vector3 translation )</code>
<code>Vector3</code>	<code>get_translation () const</code>
void	<code>set_rotation ( Vector3 rotation )</code>
<code>Vector3</code>	<code>get_rotation () const</code>
void	<code>set_scale ( Vector3 scale )</code>
<code>Vector3</code>	<code>get_scale () const</code>
void	<code>set_global_transform ( Transform global )</code>

Continued on next page

Table 9.27 – continued from previous page

<i>Transform</i>	<code>get_global_transform() const</code>
<i>Object</i>	<code>get_parent_spatial() const</code>
<code>void</code>	<code>set_ignore_transform_notification( bool enabled )</code>
<code>void</code>	<code>set_as_toplevel( bool enable )</code>
<code>bool</code>	<code>is_set_as_toplevel() const</code>
<i>World</i>	<code>get_world() const</code>
<code>void</code>	<code>update_gizmo()</code>
<code>void</code>	<code>set_gizmo( SpatialGizmo gizmo )</code>
<i>SpatialGizmo</i>	<code>get_gizmo() const</code>
<code>void</code>	<code>show()</code>
<code>void</code>	<code>hide()</code>
<code>bool</code>	<code>is_visible() const</code>
<code>bool</code>	<code>is_hidden() const</code>
<code>void</code>	<code>set_hidden( bool hidden )</code>
<code>void</code>	<code>set_notify_local_transform( bool enable )</code>
<code>bool</code>	<code>is_local_transform_notification_enabled() const</code>
<code>void</code>	<code>rotate( Vector3 normal, float radians )</code>
<code>void</code>	<code>global_rotate( Vector3 normal, float radians )</code>
<code>void</code>	<code>rotate_x( float radians )</code>
<code>void</code>	<code>rotate_y( float radians )</code>
<code>void</code>	<code>rotate_z( float radians )</code>
<code>void</code>	<code>translate( Vector3 offset )</code>
<code>void</code>	<code>global_translate( Vector3 offset )</code>
<code>void</code>	<code>orthonormalize()</code>
<code>void</code>	<code>set_identity()</code>
<code>void</code>	<code>look_at( Vector3 target, Vector3 up )</code>
<code>void</code>	<code>look_at_from_pos( Vector3 pos, Vector3 target, Vector3 up )</code>

### 9.279.3 Signals

- `visibility_changed()`

### 9.279.4 Numeric Constants

- **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.
- **NOTIFICATION\_ENTER\_WORLD = 41**
- **NOTIFICATION\_EXIT\_WORLD = 42**
- **NOTIFICATION\_VISIBILITY\_CHANGED = 43**

### 9.279.5 Description

Spatial is the base for every type of 3D *Node*. It contains a 3D *Transform* which can be set or get as local or global. If a Spatial *Node* has Spatial children, their transforms will be relative to the parent.

### 9.279.6 Member Function Description

- `void set_transform( Transform local )`

Set the transform locally, relative to the parent spatial node.

- `Transform get_transform () const`

Return the local transform, relative to the bone parent.

- `void set_translation ( Vector3 translation )`
- `Vector3 get_translation () const`
- `void set_rotation ( Vector3 rotation )`
- `Vector3 get_rotation () const`
- `void set_scale ( Vector3 scale )`
- `Vector3 get_scale () const`
- `void set_global_transform ( Transform global )`

Set the transform globally, relative to worldspace.

- `Transform get_global_transform () const`

Return the gloal 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*.

- `void set_ignore_transform_notification ( bool enabled )`
- `void set_as_toplevel ( bool enable )`
- `bool is_set_as_toplevel () const`
- `World get_world () const`
- `void update_gizmo ()`
- `void set_gizmo ( SpatialGizmo gizmo )`
- `SpatialGizmo get_gizmo () const`
- `void show ()`
- `void hide ()`
- `bool is_visible () const`
- `bool is_hidden () const`
- `void set_hidden ( bool hidden )`
- `void set_notify_local_transform ( bool enable )`
- `bool is_local_transform_notification_enabled () const`
- `void rotate ( Vector3 normal, float radians )`
- `void global_rotate ( Vector3 normal, float radians )`
- `void rotate_x ( float radians )`
- `void rotate_y ( float radians )`
- `void rotate_z ( float radians )`
- `void translate ( Vector3 offset )`
- `void global_translate ( Vector3 offset )`

- void **orthonormalize** ()
- void **set\_identity** ()
- void **look\_at** ( *Vector3* target, *Vector3* up )
- void **look\_at\_from\_pos** ( *Vector3* pos, *Vector3* target, *Vector3* up )

## 9.280 SpatialPlayer

**Inherits:** *Spatial* < *Node* < *Object*

**Inherited By:** *SpatialStreamPlayer*, *SpatialSamplePlayer*

**Category:** Core

### 9.280.1 Brief Description

### 9.280.2 Member Functions

void	<i>set_param</i> ( <i>int</i> param, <i>float</i> value )
<i>float</i>	<i>get_param</i> ( <i>int</i> param ) const

### 9.280.3 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

### 9.280.4 Member Function Description

- void *set\_param* ( *int* param, *float* value )
- *float* *get\_param* ( *int* param ) const

## 9.281 SpatialSamplePlayer

**Inherits:** *SpatialPlayer* < *Spatial* < *Node* < *Object*

**Category:** Core

## 9.281.1 Brief Description

## 9.281.2 Member Functions

void	<code>set_sample_library ( SampleLibrary library )</code>
<i>SampleLibrary</i>	<code>get_sample_library () const</code>
void	<code>set_polyphony ( int voices )</code>
<i>int</i>	<code>get_polyphony () const</code>
<i>int</i>	<code>play ( String sample, int voice=-2 )</code>
void	<code>voice_set_pitch_scale ( int voice, float ratio )</code>
void	<code>voice_set_volume_scale_db ( int voice, float db )</code>
<i>bool</i>	<code>is_voice_active ( int voice ) const</code>
void	<code>stop_voice ( int voice )</code>
void	<code>stop_all ()</code>

## 9.281.3 Numeric Constants

- `INVALID_VOICE = -1`
- `NEXT_VOICE = -2`

## 9.281.4 Member Function Description

- void `set_sample_library ( SampleLibrary library )`
- *SampleLibrary* `get_sample_library () const`
- void `set_polyphony ( int voices )`
- *int* `get_polyphony () const`
- *int* `play ( String sample, int voice=-2 )`
- void `voice_set_pitch_scale ( int voice, float ratio )`
- void `voice_set_volume_scale_db ( int voice, float db )`
- *bool* `is_voice_active ( int voice ) const`
- void `stop_voice ( int voice )`
- void `stop_all ()`

## 9.282 SpatialSound2DServer

Inherits: *Object*

Inherited By: *SpatialSound2DServerSW*

Category: Core

## 9.282.1 Brief Description

Server for Spatial 2D Sound.

### 9.282.2 Description

Server for Spatial 2D Sound.

## 9.283 SpatialSound2DServerSW

**Inherits:** *SpatialSound2DServer < Object*

**Category:** Core

### 9.283.1 Brief Description

## 9.284 SpatialSoundServer

**Inherits:** *Object*

**Inherited By:** *SpatialSoundServerSW*

**Category:** Core

### 9.284.1 Brief Description

## 9.285 SpatialSoundServerSW

**Inherits:** *SpatialSoundServer < Object*

**Category:** Core

### 9.285.1 Brief Description

## 9.286 SpatialStreamPlayer

**Inherits:** *SpatialPlayer < Spatial < Node < Object*

**Category:** Core

## 9.286.1 Brief Description

## 9.286.2 Member Functions

void	<code>set_stream ( Stream stream )</code>
Stream	<code>get_stream ( ) const</code>
void	<code>play ( float offset=0 )</code>
void	<code>stop ( )</code>
bool	<code>is_playing ( ) const</code>
void	<code>set_paused ( bool paused )</code>
bool	<code>is_paused ( ) const</code>
void	<code>set_loop ( bool enabled )</code>
bool	<code>has_loop ( ) const</code>
void	<code>set_volume ( float volume )</code>
float	<code>get_volume ( ) const</code>
void	<code>set_volume_db ( float db )</code>
float	<code>get_volume_db ( ) const</code>
void	<code>set_buffering_msec ( int msec )</code>
int	<code>get_buffering_msec ( ) const</code>
void	<code>set_loop_restart_time ( float secs )</code>
float	<code>get_loop_restart_time ( ) const</code>
String	<code>get_stream_name ( ) const</code>
int	<code>get_loop_count ( ) const</code>
float	<code>get_pos ( ) const</code>
void	<code>seek_pos ( float time )</code>
void	<code>set_autoplay ( bool enabled )</code>
bool	<code>has_autoplay ( ) const</code>
float	<code>get_length ( ) const</code>

## 9.286.3 Member Function Description

- void `set_stream ( Stream stream )`
- Stream `get_stream ( ) const`
- void `play ( float offset=0 )`
- void `stop ( )`
- bool `is_playing ( ) const`
- void `set_paused ( bool paused )`
- bool `is_paused ( ) const`
- void `set_loop ( bool enabled )`
- bool `has_loop ( ) const`
- void `set_volume ( float volume )`
- float `get_volume ( ) const`
- void `set_volume_db ( float db )`
- float `get_volume_db ( ) const`
- void `set_buffering_msec ( int msec )`

- `int get_buffering_msec () const`
- `void set_loop_restart_time ( float secs )`
- `float get_loop_restart_time () const`
- `String get_stream_name () const`
- `int get_loop_count () const`
- `float get_pos () const`
- `void seek_pos ( float time )`
- `void set_autoplay ( bool enabled )`
- `bool has_autoplay () const`
- `float get_length () const`

## 9.287 SphereShape

**Inherits:** `Shape < Resource < Reference < Object`

**Category:** Core

### 9.287.1 Brief Description

### 9.287.2 Member Functions

<code>void</code>	<code>set_radius ( float radius )</code>
<code>float</code>	<code>get_radius () const</code>

### 9.287.3 Member Function Description

- `void set_radius ( float radius )`
- `float get_radius () const`

## 9.288 SpinBox

**Inherits:** `Range < Control < CanvasItem < Node < Object`

**Category:** Core

### 9.288.1 Brief Description

Numerical input text field.

## 9.288.2 Member Functions

void	<code>set_suffix ( String suffix )</code>
<i>String</i>	<code>get_suffix () const</code>
void	<code>set_prefix ( String prefix )</code>
<i>String</i>	<code>get_prefix () const</code>
void	<code>set_editable ( bool editable )</code>
<i>bool</i>	<code>is_editable () const</code>
<i>Object</i>	<code>get_line_edit ()</code>

## 9.288.3 Description

SpinBox is a numerical input text field. It allows entering integers and floats.

## 9.288.4 Member Function Description

- void `set_suffix ( String suffix )`

Set a specific suffix.

- `String get_suffix () const`

Return the specific suffix.

- void `set_prefix ( String prefix )`

Set a prefix.

- `String get_prefix () const`

- void `set_editable ( bool editable )`

Set whether the spinbox is editable.

- `bool is_editable () const`

Return if the spinbox is editable.

- `Object get_line_edit ()`

## 9.289 SplitContainer

**Inherits:** `Container < Control < CanvasItem < Node < Object`

**Inherited By:** `HSplitContainer, VSplitContainer`

**Category:** Core

## 9.289.1 Brief Description

Container for splitting and adjusting.

## 9.289.2 Member Functions

void	<code>set_split_offset ( int offset )</code>
<i>int</i>	<code>get_split_offset () const</code>
void	<code>set_collapsed ( bool collapsed )</code>
<i>bool</i>	<code>is_collapsed () const</code>
void	<code>set_dragger_visibility ( int mode )</code>
<i>int</i>	<code>get_dragger_visibility () const</code>

## 9.289.3 Signals

- `dragged ( int offset )`

## 9.289.4 Numeric Constants

- `DRAGGER_VISIBLE = 0`
- `DRAGGER_HIDDEN = 1`
- `DRAGGER_HIDDEN_COLLAPSED = 2`

## 9.289.5 Description

Container for splitting two controls vertically or horizontally, with a grabber that allows adjusting the split offset or ratio.

## 9.289.6 Member Function Description

- void `set_split_offset ( int offset )`

Set the split offset.

- *int* `get_split_offset () const`

Return the split offset.

- void `set_collapsed ( bool collapsed )`

Set if the split must be collapsed.

- *bool* `is_collapsed () const`

Return if the split is collapsed.

- void `set_dragger_visibility ( int mode )`

- *int* `get_dragger_visibility () const`

## 9.290 SpotLight

**Inherits:** `Light < VisualInstance < Spatial < Node < Object`

**Category:** Core

## 9.290.1 Brief Description

Spotlight *Light*, such as a reflector spotlight or a latern.

## 9.290.2 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.

## 9.291 Sprite

**Inherits:** *Node2D < CanvasItem < Node < Object*

**Category:** Core

### 9.291.1 Brief Description

General purpose Sprite node.

### 9.291.2 Member Functions

void	<i>set_texture</i> ( <i>Texture</i> texture )
<i>Texture</i>	<i>get_texture</i> ( ) const
void	<i>set_centered</i> ( <i>bool</i> centered )
<i>bool</i>	<i>is_centered</i> ( ) const
void	<i>set_offset</i> ( <i>Vector2</i> offset )
<i>Vector2</i>	<i>get_offset</i> ( ) const
void	<i>set_flip_h</i> ( <i>bool</i> flip_h )
<i>bool</i>	<i>is_flipped_h</i> ( ) const
void	<i>set_flip_v</i> ( <i>bool</i> flip_v )
<i>bool</i>	<i>is_flipped_v</i> ( ) const
void	<i>set_region</i> ( <i>bool</i> enabled )
<i>bool</i>	<i>is_region</i> ( ) const
void	<i>set_region_rect</i> ( <i>Rect2</i> rect )
<i>Rect2</i>	<i>get_region_rect</i> ( ) const
void	<i>set_frame</i> ( <i>int</i> frame )
<i>int</i>	<i>get_frame</i> ( ) const
void	<i>set_vframes</i> ( <i>int</i> vframes )
<i>int</i>	<i>get_vframes</i> ( ) const
void	<i>set_hframes</i> ( <i>int</i> hframes )
<i>int</i>	<i>get_hframes</i> ( ) const
void	<i>set_modulate</i> ( <i>Color</i> modulate )
<i>Color</i>	<i>get_modulate</i> ( ) const

### 9.291.3 Signals

- **frame\_changed ()**

## 9.291.4 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 be referenced, like an atlas.

## 9.291.5 Member Function Description

- `void set_texture ( Texture texture )`

Set the base texture for the sprite.

- `Texture get_texture ( ) const`

Return the base texture for the sprite.

- `void set_centered ( bool centered )`

Set whether the sprite should be centered on the origin.

- `bool is_centered ( ) const`

Return if the sprite is centered at the local origin.

- `void set_offset ( Vector2 offset )`

Set the sprite draw offset, useful for setting rotation pivots.

- `Vector2 get_offset ( ) const`

Return sprite draw offset.

- `void set_flip_h ( bool flip_h )`

Set true to flip the sprite horizontally.

- `bool is_flipped_h ( ) const`

Return true if the sprite is flipped horizontally.

- `void set_flip_v ( bool flip_v )`

Set true to flip the sprite vertically.

- `bool is_flipped_v ( ) const`

Return true if the sprite is flipped vertically.

- `void set_region ( bool enabled )`

Set the sprite as a sub-region of a bigger texture. Useful for texture-atlases.

- `bool is_region ( ) const`

Return if the sprite reads from a region.

- `void set_region_rect ( Rect2 rect )`

Set the region rect to read from.

- `Rect2 get_region_rect ( ) const`

Return the region rect to read from.

- `void set_frame ( int frame )`

Set the texture frame for a sprite-sheet, works when vframes or hframes are greater than 1.

- `int get_frame ( ) const`

Return the texture frame for a sprite-sheet, works when vframes or hframes are greater than 1.

- `void set_vframes ( int vframes )`

Set the amount of vertical frames and converts the sprite into a sprite-sheet. This is useful for animation.

- `int get_vframes () const`

Return the amount of vertical frames. See `set_vframes`.

- `void set_hframes ( int hframes )`

Set the amount of horizontal frames and converts the sprite into a sprite-sheet. This is useful for animation.

- `int get_hframes () const`

Return the amount of horizontal frames. See `set_hframes`.

- `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.

- `Color get_modulate () const`

Return color modulation for the sprite. All sprite pixels are multiplied by this color.

## 9.292 Sprite3D

**Inherits:** `SpriteBase3D < GeometryInstance < VisualInstance < Spatial < Node < Object`

**Category:** Core

### 9.292.1 Brief Description

### 9.292.2 Member Functions

<code>void</code>	<code>set_texture ( Texture texture )</code>
<code>Texture</code>	<code>get_texture () const</code>
<code>void</code>	<code>set_region ( bool enabled )</code>
<code>bool</code>	<code>is_region () const</code>
<code>void</code>	<code>set_region_rect ( Rect2 rect )</code>
<code>Rect2</code>	<code>get_region_rect () const</code>
<code>void</code>	<code>set_frame ( int frame )</code>
<code>int</code>	<code>get_frame () const</code>
<code>void</code>	<code>set_vframes ( int vframes )</code>
<code>int</code>	<code>get_vframes () const</code>
<code>void</code>	<code>set_hframes ( int hframes )</code>
<code>int</code>	<code>get_hframes () const</code>

### 9.292.3 Signals

- `frame_changed ()`

## 9.292.4 Member Function Description

- void **set\_texture** ( *Texture* texture )
- *Texture* **get\_texture** ( ) const
- void **set\_region** ( *bool* enabled )
- *bool* **is\_region** ( ) const
- void **set\_region\_rect** ( *Rect2* rect )
- *Rect2* **get\_region\_rect** ( ) const
- void **set\_frame** ( *int* frame )
- *int* **get\_frame** ( ) const
- void **set\_vframes** ( *int* vframes )
- *int* **get\_vframes** ( ) const
- void **set\_hframes** ( *int* hframes )
- *int* **get\_hframes** ( ) const

## 9.293 SpriteBase3D

**Inherits:** *GeometryInstance* < *VisualInstance* < *Spatial* < *Node* < *Object*

**Inherited By:** *AnimatedSprite3D*, *Sprite3D*

**Category:** Core

### 9.293.1 Brief Description

### 9.293.2 Member Functions

void	<code>set_centered ( bool centered )</code>
<i>bool</i>	<code>is_centered ( ) const</code>
void	<code>set_offset ( Vector2 offset )</code>
<i>Vector2</i>	<code>get_offset ( ) const</code>
void	<code>set_flip_h ( bool flip_h )</code>
<i>bool</i>	<code>is_flipped_h ( ) const</code>
void	<code>set_flip_v ( bool flip_v )</code>
<i>bool</i>	<code>is_flipped_v ( ) const</code>
void	<code>set_modulate ( Color modulate )</code>
<i>Color</i>	<code>get_modulate ( ) const</code>
void	<code>set_opacity ( float opacity )</code>
<i>float</i>	<code>get_opacity ( ) const</code>
void	<code>set_pixel_size ( float pixel_size )</code>
<i>float</i>	<code>get_pixel_size ( ) const</code>
void	<code>set_axis ( int axis )</code>
<i>int</i>	<code>get_axis ( ) const</code>
void	<code>set_draw_flag ( int flag, bool enabled )</code>
<i>bool</i>	<code>get_draw_flag ( int flag ) const</code>
void	<code>set_alpha_cut_mode ( int mode )</code>
<i>int</i>	<code>get_alpha_cut_mode ( ) const</code>
<i>Rect2</i>	<code>get_item_rect ( ) const</code>

### 9.293.3 Numeric Constants

- **FLAG\_TRANSPARENT = 0**
- **FLAG\_SHADED = 1**
- **FLAG\_MAX = 2**
- **ALPHA\_CUT\_DISABLED = 0**
- **ALPHA\_CUT\_DISCARD = 1**
- **ALPHA\_CUT\_OPAQUE\_PREPASS = 2**

### 9.293.4 Member Function Description

- void `set_centered ( bool centered )`
- *bool* `is_centered ( ) const`
- void `set_offset ( Vector2 offset )`
- *Vector2* `get_offset ( ) const`
- void `set_flip_h ( bool flip_h )`
- *bool* `is_flipped_h ( ) const`
- void `set_flip_v ( bool flip_v )`
- *bool* `is_flipped_v ( ) const`

- void **set\_modulate** ( *Color* modulate )
- *Color* **get\_modulate** ( ) const
- void **set\_opacity** ( *float* opacity )
- *float* **get\_opacity** ( ) const
- void **set\_pixel\_size** ( *float* pixel\_size )
- *float* **get\_pixel\_size** ( ) const
- void **set\_axis** ( *int* axis )
- *int* **get\_axis** ( ) const
- void **set\_draw\_flag** ( *int* flag, *bool* enabled )
- *bool* **get\_draw\_flag** ( *int* flag ) const
- void **set\_alpha\_cut\_mode** ( *int* mode )
- *int* **get\_alpha\_cut\_mode** ( ) const
- *Rect2* **get\_item\_rect** ( ) const

## 9.294 SpriteFrames

Inherits: *Resource* < *Reference* < *Object*

Category: Core

### 9.294.1 Brief Description

Sprite frame library for *AnimatedSprite*.

### 9.294.2 Member Functions

void	<i>add_frame</i> ( <i>Object</i> frame, <i>int</i> atpos=-1 )
<i>int</i>	<i>get_frame_count</i> ( ) const
<i>Object</i>	<i>get_frame</i> ( <i>int</i> idx ) const
void	<i>set_frame</i> ( <i>int</i> idx, <i>Object</i> txt )
void	<i>remove_frame</i> ( <i>int</i> idx )
void	<i>clear</i> ( )

### 9.294.3 Description

Sprite frame library for *AnimatedSprite*.

### 9.294.4 Member Function Description

- void **add\_frame** ( *Object* frame, *int* atpos=-1 )

Add a frame (texture).

- *int* **get\_frame\_count** ( ) const

Return the amount of frames.

- `Object get_frame ( int idx ) const`

Return a texture (frame).

- `void set_frame ( int idx, Object txt )`
- `void remove_frame ( int idx )`

Remove a frame

- `void clear ()`

Clear the frames.

## 9.295 StaticBody

**Inherits:** `PhysicsBody < CollisionObject < Spatial < Node < Object`

**Category:** Core

### 9.295.1 Brief Description

PhysicsBody for static collision objects.

### 9.295.2 Member Functions

<code>void</code>	<code>set_constant_linear_velocity ( Vector3 vel )</code>
<code>void</code>	<code>set_constant_angular_velocity ( Vector3 vel )</code>
<code>Vector3</code>	<code>get_constant_linear_velocity () const</code>
<code>Vector3</code>	<code>get_constant_angular_velocity () const</code>
<code>void</code>	<code>set_friction ( float friction )</code>
<code>float</code>	<code>get_friction () const</code>
<code>void</code>	<code>set_bounce ( float bounce )</code>
<code>float</code>	<code>get_bounce () const</code>

### 9.295.3 Description

StaticBody implements a static collision `Node`, by utilizing a rigid body in the `PhysicsServer`. Static bodies are used for static collision. For more information on physics body nodes, see `PhysicsBody`.

### 9.295.4 Member Function Description

- `void set_constant_linear_velocity ( Vector3 vel )`
- `void set_constant_angular_velocity ( Vector3 vel )`
- `Vector3 get_constant_linear_velocity () const`
- `Vector3 get_constant_angular_velocity () const`
- `void set_friction ( float friction )`
- `float get_friction () const`

- void `set_bounce` (`float` bounce )
- `float get_bounce () const`

## 9.296 StaticBody2D

**Inherits:** `PhysicsBody2D < CollisionObject2D < Node2D < CanvasItem < Node < Object`

**Category:** Core

### 9.296.1 Brief Description

Static body for 2D Physics.

### 9.296.2 Member Functions

void	<code>set_constant_linear_velocity</code> ( <code>Vector2</code> vel )
void	<code>set_constant_angular_velocity</code> ( <code>float</code> vel )
<code>Vector2</code>	<code>get_constant_linear_velocity () const</code>
<code>float</code>	<code>get_constant_angular_velocity () const</code>
void	<code>set_friction</code> ( <code>float</code> friction )
<code>float</code>	<code>get_friction () const</code>
void	<code>set_bounce</code> ( <code>float</code> bounce )
<code>float</code>	<code>get_bounce () const</code>

### 9.296.3 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).

### 9.296.4 Member Function Description

- 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.

- void `set_constant_angular_velocity` (`float` vel )

Set a constant angular velocity for the body.

- `Vector2 get_constant_linear_velocity () const`

Return the constant linear velocity for the body. This does not rotate the body, but affects other bodies touching it, as if it was rotating.

- `float get_constant_angular_velocity () const`

Return the constant angular velocity for the body.

- `void set_friction (float friction)`

Set the body friction, from 0 (frictionless) to 1 (full friction).

- `float get_friction () const`

Return the body friction.

- `void set_bounce (float bounce)`

Set the body bounciness, from 0 (not bouncy) to 1 (bouncy).

- `float get_bounce () const`

Return the body bounciness.

## 9.297 StreamPeer

**Inherits:** [Reference](#) < [Object](#)

**Inherited By:** [StreamPeerSSL](#), [StreamPeerTCP](#)

**Category:** Core

### 9.297.1 Brief Description

Abstraction and base class for stream-based protocols.

### 9.297.2 Member Functions

<code>int</code>	<code>put_data (RawArray data)</code>
<code>Array</code>	<code>put_partial_data (RawArray data)</code>
<code>Array</code>	<code>get_data (int bytes)</code>
<code>Array</code>	<code>get_partial_data (int bytes)</code>
<code>int</code>	<code>get_available_bytes () const</code>
<code>void</code>	<code>set_big_endian (bool enable)</code>
<code>bool</code>	<code>is_big_endian_enabled () const</code>
<code>void</code>	<code>put_8 (int val)</code>
<code>void</code>	<code>put_u8 (int val)</code>
<code>void</code>	<code>put_16 (int val)</code>
<code>void</code>	<code>put_u16 (int val)</code>
<code>void</code>	<code>put_32 (int val)</code>
<code>void</code>	<code>put_u32 (int val)</code>
<code>void</code>	<code>put_64 (int val)</code>
<code>void</code>	<code>put_u64 (int val)</code>
<code>void</code>	<code>put_float (float val)</code>
<code>void</code>	<code>put_double (float val)</code>
<code>void</code>	<code>put_utf8_string (String val)</code>
<code>void</code>	<code>put_var (Variant val)</code>
Continued on next page	

Table 9.28 – continued from previous page

<i>int</i>	<code>get_8()</code>
<i>int</i>	<code>get_u8()</code>
<i>int</i>	<code>get_16()</code>
<i>int</i>	<code>get_u16()</code>
<i>int</i>	<code>get_32()</code>
<i>int</i>	<code>get_u32()</code>
<i>int</i>	<code>get_64()</code>
<i>int</i>	<code>get_u64()</code>
<i>float</i>	<code>get_float()</code>
<i>float</i>	<code>get_double()</code>
<i>String</i>	<code>get_string( int bytes )</code>
<i>String</i>	<code>get_utf8_string( int bytes )</code>
Variant	<code>get_var()</code>

### 9.297.3 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.

### 9.297.4 Member Function Description

- `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.

- `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.

- `Array get_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 block until the desired amount is received. This function returns two values, an Error code and a data array.

- `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.

- `int get_available_bytes() const`

- `void set_big_endian( bool enable )`

- `bool is_big_endian_enabled() const`

- `void put_8( int val )`

- `void put_u8( int val )`

- `void put_16( int val )`

- `void put_u16( int val )`

- `void put_32( int val )`

- void **put\_u32** (*int* val)
- void **put\_64** (*int* val)
- void **put\_u64** (*int* val)
- void **put\_float** (*float* val)
- void **put\_double** (*float* val)
- void **put\_utf8\_string** (*String* val)
- void **put\_var** (Variant val)
- *int* **get\_8** ()
- *int* **get\_u8** ()
- *int* **get\_16** ()
- *int* **get\_u16** ()
- *int* **get\_32** ()
- *int* **get\_u32** ()
- *int* **get\_64** ()
- *int* **get\_u64** ()
- *float* **get\_float** ()
- *float* **get\_double** ()
- *String* **get\_string** (*int* bytes)
- *String* **get\_utf8\_string** (*int* bytes)
- Variant **get\_var** ()

## 9.298 StreamPeerSSL

**Inherits:** *StreamPeer* < *Reference* < *Object*

**Category:** Core

### 9.298.1 Brief Description

### 9.298.2 Member Functions

Error	<i>accept</i> ( <i>StreamPeer</i> stream)
Error	<i>connect</i> ( <i>StreamPeer</i> stream, <i>bool</i> validate_certs=false, <i>String</i> for_hostname="" )
<i>int</i>	<i>get_status</i> () const
void	<i>disconnect</i> ()

### 9.298.3 Numeric Constants

- **STATUS\_DISCONNECTED = 0**
- **STATUS\_CONNECTED = 1**

- STATUS\_ERROR\_NO\_CERTIFICATE = 2
- STATUS\_ERROR\_HOSTNAME\_MISMATCH = 3

## 9.298.4 Member Function Description

- Error **accept** ( *StreamPeer* stream )
- Error **connect** ( *StreamPeer* stream, *bool* validate\_certs=false, *String* for\_hostname="" )
- *int* **get\_status** ( ) const
- void **disconnect** ( )

## 9.299 StreamPeerTCP

Inherits: *StreamPeer* < *Reference* < *Object*

Category: Core

### 9.299.1 Brief Description

TCP Stream peer.

### 9.299.2 Member Functions

<i>int</i>	<i>connect</i> ( <i>String</i> host, <i>int</i> port )
<i>bool</i>	<i>is_connected</i> ( ) const
<i>int</i>	<i>get_status</i> ( ) const
<i>String</i>	<i>get_connected_host</i> ( ) const
<i>int</i>	<i>get_connected_port</i> ( ) const
void	<i>disconnect</i> ( )

### 9.299.3 Numeric Constants

- STATUS\_NONE = 0
- STATUS\_CONNECTING = 1
- STATUS\_CONNECTED = 2
- STATUS\_ERROR = 3

### 9.299.4 Description

TCP Stream peer. This object can be used to connect to TCP servers, or also is returned by a tcp server.

## 9.299.5 Member Function Description

- `int connect ( String host, int port )`
- `bool is_connected () const`
- `int get_status () const`
- `String get_connected_host () const`
- `int get_connected_port () const`
- `void disconnect ()`

## 9.300 StreamPlayer

**Inherits:** `Node < Object`

**Category:** Core

### 9.300.1 Brief Description

Base class for audio stream playback.

### 9.300.2 Member Functions

<code>void</code>	<code>set_stream ( Stream stream )</code>
<code>Stream</code>	<code>get_stream () const</code>
<code>void</code>	<code>play ( float offset=0 )</code>
<code>void</code>	<code>stop ()</code>
<code>bool</code>	<code>is_playing () const</code>
<code>void</code>	<code>set_paused ( bool paused )</code>
<code>bool</code>	<code>is_paused () const</code>
<code>void</code>	<code>set_loop ( bool enabled )</code>
<code>bool</code>	<code>has_loop () const</code>
<code>void</code>	<code>set_volume ( float volume )</code>
<code>float</code>	<code>get_volume () const</code>
<code>void</code>	<code>set_volume_db ( float db )</code>
<code>float</code>	<code>get_volume_db () const</code>
<code>void</code>	<code>set_buffering_msec ( int msec )</code>
<code>int</code>	<code>get_buffering_msec () const</code>
<code>void</code>	<code>set_loop_restart_time ( float secs )</code>
<code>float</code>	<code>get_loop_restart_time () const</code>
<code>String</code>	<code>get_stream_name () const</code>
<code>int</code>	<code>get_loop_count () const</code>
<code>float</code>	<code>get_pos () const</code>
<code>void</code>	<code>seek_pos ( float time )</code>
<code>void</code>	<code>set_autoplay ( bool enabled )</code>
<code>bool</code>	<code>has_autoplay () const</code>
<code>float</code>	<code>get_length () const</code>

### 9.300.3 Signals

- **finished ()**

### 9.300.4 Description

Base class for audio stream playback. Audio stream players inherit from it.

### 9.300.5 Member Function Description

- void **set\_stream ( Stream stream )**

Set the *EventStream* this player will play.

- Stream **get\_stream () const**

Return the currently assigned stream.

- void **play ( float offset=0 )**

Play the currently assigned stream, starting from a given position (in seconds).

- void **stop ()**

Stop the playback.

- bool **is\_playing () const**

Return whether this player is playing.

- void **set\_paused ( bool paused )**

Pause stream playback.

- bool **is\_paused () const**

Return whether the playback is currently paused.

- void **set\_loop ( bool enabled )**

Set whether the stream will be restarted at the end.

- bool **has\_loop () const**

Return whether the stream will be restarted at the end.

- 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.

- float **get\_volume () const**

Return the playback volume for this player.

- 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.

- float **get\_volume\_db () const**

Return the playback volume for this player, in decibels.

- `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.

- `int get_buffering_msec () const`

Return the size of the audio buffer.

- `void set_loop_restart_time ( float secs )`

Set the point in time the stream will rewind to, when looping.

- `float get_loop_restart_time () const`

Return the point in time the stream will rewind to, when looping.

- `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>”.

- `int get_loop_count () const`

Return the number of times the playback has looped.

- `float get_pos () const`

Return the playback 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.

- `bool has_autoplay () const`

Return whether this player will start playing as soon as it enters the scene tree.

- `float get_length () const`

Return the length of the stream, in seconds.

## 9.301 String

**Category:** Built-In Types

### 9.301.1 Brief Description

Built-in string class.

Continued on next page

Table 9.29 – continued from previous page

### 9.301.2 Member Functions

<i>String</i>	<i>basename ()</i>
<i>bool</i>	<i>begins_with ( String text )</i>
<i>String</i>	<i>c_escape ()</i>
<i>String</i>	<i>c_unescape ()</i>
<i>String</i>	<i>capitalize ()</i>
<i>int</i>	<i>casecmp_to ( String to )</i>
<i>bool</i>	<i>empty ()</i>
<i>String</i>	<i>extension ()</i>
<i>int</i>	<i>find ( String what, int from=0 )</i>
<i>int</i>	<i>find_last ( String what )</i>
<i>int</i>	<i>findn ( String what, int from=0 )</i>
<i>String</i>	<i>get_base_dir ()</i>
<i>String</i>	<i>get_file ()</i>
<i>int</i>	<i>hash ()</i>
<i>int</i>	<i>hex_to_int ()</i>
<i>String</i>	<i>insert ( int pos, String what )</i>
<i>bool</i>	<i>is_abs_path ()</i>
<i>bool</i>	<i>is_rel_path ()</i>
<i>bool</i>	<i>is_valid_float ()</i>
<i>bool</i>	<i>is_valid_html_color ()</i>
<i>bool</i>	<i>is_valid_identifier ()</i>
<i>bool</i>	<i>is_valid_integer ()</i>
<i>bool</i>	<i>is_valid_ip_address ()</i>
<i>String</i>	<i>json_escape ()</i>
<i>String</i>	<i>left ( int pos )</i>
<i>int</i>	<i>length ()</i>
<i>bool</i>	<i>match ( String expr )</i>
<i>bool</i>	<i>matchn ( String expr )</i>
<i>RawArray</i>	<i>md5_buffer ()</i>
<i>String</i>	<i>md5_text ()</i>
<i>int</i>	<i>nocasecmp_to ( String to )</i>
<i>String</i>	<i>ord_at ( int at )</i>
<i>String</i>	<i>pad_decimals ( int digits )</i>
<i>String</i>	<i>pad_zeros ( int digits )</i>
<i>String</i>	<i>percent_decode ()</i>
<i>String</i>	<i>percent_encode ()</i>
<i>String</i>	<i>plus_file ( String file )</i>
<i>String</i>	<i>replace ( String what, String forwhat )</i>
<i>String</i>	<i>replacen ( String what, String forwhat )</i>
<i>int</i>	<i>rfind ( String what, int from=-1 )</i>
<i>int</i>	<i>rfindn ( String what, int from=-1 )</i>
<i>String</i>	<i>right ( int pos )</i>
<i>StringArray</i>	<i>split ( String divisor, bool allow_empty=True )</i>
<i>RealArray</i>	<i>split_floats ( String divisor, bool allow_empty=True )</i>
<i>String</i>	<i>strip_edges ()</i>
<i>String</i>	<i>substr ( int from, int len )</i>

Continued on next page

Table 9.29 – continued from previous page

<code>RawArray</code>	<code>to_ascii()</code>
<code>float</code>	<code>to_float()</code>
<code>int</code>	<code>to_int()</code>
<code>String</code>	<code>to_lower()</code>
<code>String</code>	<code>to_upper()</code>
<code>RawArray</code>	<code>to_utf8()</code>
<code>String</code>	<code>xml_escape()</code>
<code>String</code>	<code>xml_unescape()</code>

### 9.301.3 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.

### 9.301.4 Member Function Description

- `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.

- `String c_escape()`
- `String c_unescape()`
- `String capitalize()`

Return the string in uppercase.

- `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.

- `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 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_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()`

- `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()`

- `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 matching, using ? and \* wildcards.

- `bool matchn( String expr )`

Do a simple, case insensitive, expression matching, using ? and \* wildcards.

- `RawArray md5_buffer()`

- `String md5_text()`

- `int nocasecmp_to( String to )`

Perform a case-insensitive comparison to another string, return -1 if less, 0 if equal and +1 if greater.

- `String ord_at( int at )`

Return the character code at position “at”.

- *String* **pad\_decimals** ( *int* digits )
- *String* **pad\_zeros** ( *int* digits )
- *String* **percent\_decode** ( )
- *String* **percent\_encode** ( )
- *String* **plus\_file** ( *String* file )
- *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.

- *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 :ref:“One”,“Two”,“Three”<class\_“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 :ref:“1,2.5,3”<class\_“1,2.5,3”> if split by “,”.

- *String* **strip\_edges** ( )

Return a copy of the string stripped of any non-printable character at the beginning and the end.

- *String* **substr** ( *int* from, *int* len )

Return part of the string from “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()`

Perform XML escaping on the string.

- `String xml_unescape()`

Perform XML un-escaping of the string.

## 9.302 **StringArray**

**Category:** Built-In Types

### 9.302.1 Brief Description

String Array.

### 9.302.2 Member Functions

void	<code>push_back ( String string )</code>
void	<code>resize ( int idx )</code>
void	<code>set ( int idx, String string )</code>
<code>int</code>	<code>size ()</code>
<code>StringArray</code>	<code>StringArray ( Array from )</code>

### 9.302.3 Description

String Array. Array of strings. Can only contain strings. Optimized for memory usage, can't fragment the memory.

### 9.302.4 Member Function Description

- `void push_back ( String string )`
- `void resize ( int idx )`
- `void set ( int idx, String string )`
- `int size ()`
- `StringArray StringArray ( Array from )`

## 9.303 StyleBox

**Inherits:** [Resource](#) < [Reference](#) < [Object](#)

**Inherited By:** [StyleBoxImageMask](#), [StyleBoxFlat](#), [StyleBoxTexture](#), [StyleBoxEmpty](#)

**Category:** Core

### 9.303.1 Brief Description

Base class for drawing stylized boxes for the UI.

### 9.303.2 Member Functions

<code>bool</code>	<code>test_mask ( Vector2 point, Rect2 rect ) const</code>
<code>void</code>	<code>set_default_margin ( int margin, float offset )</code>
<code>float</code>	<code>get_default_margin ( int margin ) const</code>
<code>float</code>	<code>get_margin ( int margin ) const</code>
<code>Vector2</code>	<code>get_minimum_size () const</code>
<code>Vector2</code>	<code>get_center_size () const</code>
<code>Vector2</code>	<code>get_offset () const</code>
<code>void</code>	<code>draw ( RID canvas_item, Rect2 rect ) const</code>

### 9.303.3 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.

### 9.303.4 Member Function Description

- `bool test_mask ( Vector2 point, Rect2 rect ) const`

Test a position in a rectangle, return whether it passes the mask test.

- `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.

- `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_center_size () const`

- `Vector2 get_offset() const`

Return the “offset” of a stylebox, this is a helper function, like writing `Point2( style.get_margin(MARGIN_LEFT), style.get_margin(MARGIN_TOP) )`

- `void draw( RID canvas_item, Rect2 rect ) const`

## 9.304 StyleBoxEmpty

**Inherits:** `StyleBox < Resource < Reference < Object`

**Category:** Core

### 9.304.1 Brief Description

Empty stylebox (does not display anything).

### 9.304.2 Description

Empty stylebox (really does not display anything).

## 9.305 StyleBoxFlat

**Inherits:** `StyleBox < Resource < Reference < Object`

**Category:** Core

### 9.305.1 Brief Description

Stylebox of a single color.

### 9.305.2 Member Functions

<code>void</code>	<code>set_bg_color( Color color )</code>
<code>Color</code>	<code>get_bg_color() const</code>
<code>void</code>	<code>set_light_color( Color color )</code>
<code>Color</code>	<code>get_light_color() const</code>
<code>void</code>	<code>set_dark_color( Color color )</code>
<code>Color</code>	<code>get_dark_color() const</code>
<code>void</code>	<code>set_border_size( int size )</code>
<code>int</code>	<code>get_border_size() const</code>
<code>void</code>	<code>set_border_blend( bool blend )</code>
<code>bool</code>	<code>get_border_blend() const</code>
<code>void</code>	<code>set_draw_center( bool size )</code>
<code>bool</code>	<code>get_draw_center() const</code>

### 9.305.3 Description

Stylebox of a single color. Displays the stylebox of a single color, alternatively a border with light/dark colors can be assigned.

### 9.305.4 Member Function Description

- void `set_bg_color ( Color color )`
- `Color get_bg_color () const`
- void `set_light_color ( Color color )`
- `Color get_light_color () const`
- void `set_dark_color ( Color color )`
- `Color get_dark_color () const`
- void `set_border_size ( int size )`
- `int get_border_size () const`
- void `set_border_blend ( bool blend )`
- `bool get_border_blend () const`
- void `set_draw_center ( bool size )`
- `bool get_draw_center () const`

## 9.306 StyleBoxImageMask

**Inherits:** `StyleBox < Resource < Reference < Object`

**Category:** Core

### 9.306.1 Brief Description

Image mask based StyleBox, for mask test.

### 9.306.2 Member Functions

void	<code>set_image ( Image image )</code>
<code>Image</code>	<code>get_image () const</code>
void	<code>set_expand ( bool expand )</code>
<code>bool</code>	<code>get_expand () const</code>
void	<code>set_expand_margin_size ( int margin, float size )</code>
<code>float</code>	<code>get_expand_margin_size ( int margin ) const</code>

### 9.306.3 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.

## 9.306.4 Member Function Description

- 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.

- *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.

- *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.

- 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.

- *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.

## 9.307 StyleBoxTexture

**Inherits:** *StyleBox* < *Resource* < *Reference* < *Object*

**Category:** Core

### 9.307.1 Brief Description

Texture Based 3x3 scale style.

### 9.307.2 Member Functions

void	<i>set_texture</i> ( <i>Texture</i> texture )
<i>Texture</i>	<i>get_texture</i> ( ) const
void	<i>set_margin_size</i> ( <i>int</i> margin, <i>float</i> size )
<i>float</i>	<i>get_margin_size</i> ( <i>int</i> margin ) const
void	<i>set_expand_margin_size</i> ( <i>int</i> margin, <i>float</i> size )
<i>float</i>	<i>get_expand_margin_size</i> ( <i>int</i> margin ) const
void	<i>set_draw_center</i> ( <i>bool</i> enable )
<i>bool</i>	<i>get_draw_center</i> ( ) const

### 9.307.3 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.

## 9.307.4 Member Function Description

- void `set_texture` ( *Texture* texture )
- *Texture* `get_texture` ( ) const
- void `set_margin_size` ( *int* margin, *float* size )
- *float* `get_margin_size` ( *int* margin ) const
- void `set_expand_margin_size` ( *int* margin, *float* size )
- *float* `get_expand_margin_size` ( *int* margin ) const
- void `set_draw_center` ( *bool* enable )
- *bool* `get_draw_center` ( ) const

## 9.308 SurfaceTool

**Inherits:** *Reference* < *Object*

**Category:** Core

### 9.308.1 Brief Description

Helper tool to create geometry.

### 9.308.2 Member Functions

void	<code>begin</code> ( <i>int</i> primitive )
void	<code>add_vertex</code> ( <i>Vector3</i> vertex )
void	<code>add_color</code> ( <i>Color</i> color )
void	<code>add_normal</code> ( <i>Vector3</i> normal )
void	<code>add_tangent</code> ( <i>Plane</i> tangent )
void	<code>add_uv</code> ( <i>Vector2</i> uv )
void	<code>add_uv2</code> ( <i>Vector2</i> uv2 )
void	<code>add_bones</code> ( <i>IntArray</i> bones )
void	<code>add_weights</code> ( <i>RealArray</i> weights )
void	<code>add_smooth_group</code> ( <i>bool</i> smooth )
void	<code>set_material</code> ( <i>Material</i> material )
void	<code>index</code> ( )
void	<code>deindex</code> ( )
void	<code>generate_normals</code> ( )
<i>Mesh</i>	<code>commit</code> ( <i>Mesh</i> existing=Object() )
void	<code>clear</code> ( )

### 9.308.3 Description

Helper tool to create geometry.

## 9.308.4 Member Function Description

- void **begin** ( *int* primitive )
- void **add\_vertex** ( *Vector3* vertex )
- void **add\_color** ( *Color* color )
- void **add\_normal** ( *Vector3* normal )
- void **add\_tangent** ( *Plane* tangent )
- void **add\_uv** ( *Vector2* uv )
- void **add\_uv2** ( *Vector2* uv2 )
- void **add\_bones** ( *IntArray* bones )
- void **add\_weights** ( *RealArray* weights )
- void **add\_smooth\_group** ( *bool* smooth )
- void **set\_material** ( *Material* material )
- void **index** ( )
- void **deindex** ( )
- void **generate\_normals** ( )
- *Mesh* **commit** ( *Mesh* existing=Object() )
- void **clear** ( )

## 9.309 TabContainer

**Inherits:** *Control* < *CanvasItem* < *Node* < *Object*

**Category:** Core

### 9.309.1 Brief Description

Tabbed Container.

## 9.309.2 Member Functions

<i>int</i>	<code>get_tab_count()</code> const
<code>void</code>	<code>set_current_tab( int tab_idx )</code>
<i>int</i>	<code>get_current_tab()</code> const
<i>Control</i>	<code>get_current_tab_control()</code> const
<i>Control</i>	<code>get_tab_control( int idx )</code> const
<code>void</code>	<code>set_tab_align( int align )</code>
<i>int</i>	<code>get_tab_align()</code> const
<code>void</code>	<code>set_tabs_visible( bool visible )</code>
<i>bool</i>	<code>are_tabs_visible()</code> const
<code>void</code>	<code>set_tab_title( int tab_idx, String title )</code>
<i>String</i>	<code>get_tab_title( int tab_idx )</code> const
<code>void</code>	<code>set_tab_icon( int tab_idx, Texture icon )</code>
<i>Texture</i>	<code>get_tab_icon( int tab_idx )</code> const
<code>void</code>	<code>set_popup( Popup popup )</code>
<i>Popup</i>	<code>get_popup()</code> const

## 9.309.3 Signals

- `pre_popup_pressed()`
- `tab_changed( int tab )`

## 9.309.4 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.

## 9.309.5 Member Function Description

- `int get_tab_count()` const

Return the amount of tabs.

- `void set_current_tab( int tab_idx )`

Bring a tab (and the Control it represents) to the front, and hide the rest.

- `int get_current_tab()` const

Return the current tab that is being showed.

- `Control get_current_tab_control()` const
- `Control get_tab_control( int idx )` const
- `void set_tab_align( int align )`

Set tab alignment, from the ALIGN\_\* enum. Moves tabs to the left, right or center.

- `int get_tab_align()` const

Return tab alignment, from the ALIGN\_\* enum.

- `void set_tabs_visible( bool visible )`

Set whether the tabs should be visible or hidden.

- `bool are_tabs_visible () const`

Return whether the tabs should be visible or hidden.

- `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.

- `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_tab_icon ( int tab_idx, Texture icon )`

Set an icon for a tab.

- `Texture get_tab_icon ( int tab_idx ) const`
- `void set_popup ( Popup popup )`
- `Popup get_popup () const`

## 9.310 Tabs

**Inherits:** `Control < CanvasItem < Node < Object`

**Category:** Core

### 9.310.1 Brief Description

Tabs Control.

### 9.310.2 Member Functions

<code>int</code>	<code>get_tab_count () const</code>
<code>void</code>	<code>set_current_tab ( int tab_idx )</code>
<code>int</code>	<code>get_current_tab () const</code>
<code>void</code>	<code>set_tab_title ( int tab_idx, String title )</code>
<code>String</code>	<code>get_tab_title ( int tab_idx ) const</code>
<code>void</code>	<code>set_tab_icon ( int tab_idx, Texture icon )</code>
<code>Texture</code>	<code>get_tab_icon ( int tab_idx ) const</code>
<code>void</code>	<code>remove_tab ( int tab_idx )</code>
<code>void</code>	<code>add_tab ( String title, Texture icon )</code>
<code>void</code>	<code>set_tab_align ( int align )</code>
<code>int</code>	<code>get_tab_align () const</code>
<code>void</code>	<code>ensure_tab_visible ( int idx )</code>

### 9.310.3 Signals

- `tab_close ( int tab )`
- `right_button_pressed ( int tab )`
- `tab_changed ( int tab )`

## 9.310.4 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**

## 9.310.5 Description

Simple tabs control, similar to [TabContainer](#) but is only in charge of drawing tabs, not interact with children.

## 9.310.6 Member Function Description

- `int get_tab_count () const`
- `void set_current_tab ( int tab_idx )`
- `int get_current_tab () const`
- `void set_tab_title ( int tab_idx, String title )`
- `String get_tab_title ( int tab_idx ) const`
- `void set_tab_icon ( int tab_idx, Texture icon )`
- `Texture get_tab_icon ( int tab_idx ) const`
- `void remove_tab ( int tab_idx )`
- `void add_tab ( String title, Texture icon )`
- `void set_tab_align ( int align )`
- `int get_tab_align () const`
- `void ensure_tab_visible ( int idx )`

## 9.311 TCP\_Server

Inherits: [Reference](#) < [Object](#)

Category: Core

### 9.311.1 Brief Description

TCP Server.

### 9.311.2 Member Functions

<i>int</i>	<i>listen</i> ( <i>int</i> port, <i>StringArray</i> accepted_hosts= <i>StringArray</i> () )
<i>bool</i>	<i>is_connection_available</i> ( ) const
<i>Object</i>	<i>take_connection</i> ( )
<i>void</i>	<i>stop</i> ( )

### 9.311.3 Description

TCP Server class. Listens to connections on a port and returns a *StreamPeerTCP* when got a connection.

### 9.311.4 Member Function Description

- *int* **listen** ( *int* port, *StringArray* accepted\_hosts=*StringArray*() )

Listen on a port, alternatively give a white-list of accepted hosts.

- *bool* **is\_connection\_available** ( ) const

Return true if a connection is available for taking.

- *Object* **take\_connection** ( )

If a connection is available, return a StreamPeerTCP with the connection/

- *void* **stop** ( )

Stop listening.

## 9.312 TestCube

**Inherits:** *GeometryInstance* < *VisualInstance* < *Spatial* < *Node* < *Object*

**Category:** Core

### 9.312.1 Brief Description

## 9.313 TextEdit

**Inherits:** *Control* < *CanvasItem* < *Node* < *Object*

**Category:** Core

### 9.313.1 Brief Description

Multiline text editing control.

Continued on next page

Table 9.30 – continued from previous page

### 9.313.2 Member Functions

void	<code>set_text ( String text )</code>
void	<code>insert_text_at_cursor ( String text )</code>
<i>int</i>	<code>get_line_count () const</code>
<i>String</i>	<code>get_text ()</code>
<i>String</i>	<code>get_line ( int line ) const</code>
void	<code>cursor_set_column ( int column, bool adjust_viewport=false )</code>
void	<code>cursor_set_line ( int line, bool adjust_viewport=false )</code>
<i>int</i>	<code>cursor_get_column () const</code>
<i>int</i>	<code>cursor_get_line () const</code>
void	<code>set_READONLY ( bool enable )</code>
void	<code>set_wrap ( bool enable )</code>
void	<code>set_max_chars ( int amount )</code>
void	<code>cut ()</code>
void	<code>copy ()</code>
void	<code>paste ()</code>
void	<code>select_all ()</code>
void	<code>select ( int from_line, int from_column, int to_line, int to_column )</code>
<i>bool</i>	<code>is_selection_active () const</code>
<i>int</i>	<code>get_selection_from_line () const</code>
<i>int</i>	<code>get_selection_from_column () const</code>
<i>int</i>	<code>get_selection_to_line () const</code>
<i>int</i>	<code>get_selection_to_column () const</code>
<i>String</i>	<code>get_selection_text () const</code>
<i>String</i>	<code>get_word_under_cursor () const</code>
<i>IntArray</i>	<code>search ( String flags, int from_line, int from_column, int to_line ) const</code>
void	<code>undo ()</code>
void	<code>redo ()</code>
void	<code>clear_undo_history ()</code>
void	<code>set_syntax_coloring ( bool enable )</code>
<i>bool</i>	<code>is_syntax_coloring_enabled () const</code>
void	<code>add_keyword_color ( String keyword, Color color )</code>
void	<code>add_color_region ( String begin_key, String end_key, Color color, bool line_only=false )</code>
void	<code>set_symbol_color ( Color color )</code>
void	<code>set_custom_bg_color ( Color color )</code>
void	<code>clear_colors ()</code>

### 9.313.3 Signals

- `text_changed ()`
- `cursor_changed ()`
- `request_completion ()`

### 9.313.4 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.

### 9.313.5 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.

### 9.313.6 Member Function Description

- `void set_text ( String text )`

Set the entire text.

- `void insert_text_at_cursor ( String text )`

Insert a given text at the cursor position.

- `int get_line_count ( ) const`

Return the amount of total lines in the text.

- `String get_text ( )`

Return the whole text.

- `String get_line ( int line ) const`

Return the text of a specific line.

- `void cursor_set_column ( int column, bool adjust_viewport=false )`

- `void cursor_set_line ( int line, bool adjust_viewport=false )`

- `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.

- `void set_READONLY ( bool enable )`

Set the text editor as read-only. Text can be displayed but not edited.

- `void set_WRAP ( bool enable )`

Enable text wrapping when it goes beyond the edge of what is visible.

- `void set_MAX_CHARS ( int amount )`

Set the maximum amount of characters editable.

- `void cut ( )`

Cut the current selection.

- `void copy ( )`

Copy the current selection.

- `void paste ( )`

Paste the current selection.

- void **select\_all()**

Select all the text.

- void **select( int from\_line, int from\_column, int to\_line, int to\_column )**

Perform selection, from line/column to line/column.

- *bool* **is\_selection\_active()** const

Return true if the selection is active.

- *int* **get\_selection\_from\_line()** const

Return the selection begin line.

- *int* **get\_selection\_from\_column()** const

Return the selection begin column.

- *int* **get\_selection\_to\_line()** const

Return the selection end line.

- *int* **get\_selection\_to\_column()** const

Return the selection end column.

- *String* **get\_selection\_text()** const

Return the text inside the selection.

- *String* **get\_word\_under\_cursor()** const

- *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 **undo()**

Perform undo operation.

- void **redo()**

Perform redo operation.

- void **clear\_undo\_history()**

Clear the undo history.

- void **set\_syntax\_coloring( bool enable )**

Set to enable the syntax coloring.

- *bool* **is\_syntax\_coloring\_enabled()** const

Return true if the syntax coloring is enabled.

- void **add\_keyword\_color( String keyword, Color color )**

Add a keyword and its color.

- 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 **set\_symbol\_color( Color color )**

Set the color for symbols.

- void **set\_custom\_bg\_color( Color color )**

Set a custom background color. A background color with alpha==0 disables this.

- void **clear\_colors ()**

Clear all the syntax coloring information.

## 9.314 Texture

**Inherits:** *Resource < Reference < Object*

**Inherited By:** *RenderTargetTexture, AtlasTexture, ImageTexture, LargeTexture*

**Category:** Core

### 9.314.1 Brief Description

Texture for 2D and 3D.

### 9.314.2 Member Functions

<i>int</i>	<i>get_width () const</i>
<i>int</i>	<i>get_height () const</i>
<i>Vector2</i>	<i>get_size () const</i>
<i>RID</i>	<i>get_rid () const</i>
<i>bool</i>	<i>has_alpha () const</i>
<i>void</i>	<i>set_flags ( int flags )</i>
<i>int</i>	<i>get_flags () const</i>
<i>void</i>	<i>draw ( RID canvas_item, Vector2 pos, Color modulate=Color(1,1,1,1), bool transpose=false ) const</i>
<i>void</i>	<i>draw_rect ( RID canvas_item, Rect2 rect, bool tile, Color modulate=Color(1,1,1,1), bool transpose=false ) const</i>
<i>void</i>	<i>draw_rect_region ( RID canvas_item, Rect2 rect, Rect2 src_rect, Color modulate=Color(1,1,1,1), bool transpose=false ) const</i>

### 9.314.3 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**

## 9.314.4 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](#).

## 9.314.5 Member Function Description

- `int get_width () const`

Return the texture width.

- `int get_height () const`

Return the texture height.

- `Vector2 get_size () const`

Return the texture size.

- `RID get_rid () const`

Return the texture RID as used in the [VisualServer](#).

- `bool has_alpha () const`

- `void set_flags ( int flags )`

Change the texture flags.

- `int get_flags () const`

Return the current texture flags.

- `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`

## 9.315 TextureButton

**Inherits:** [BaseButton](#) < [Control](#) < [CanvasItem](#) < [Node](#) < [Object](#)

**Category:** Core

### 9.315.1 Brief Description

Button that can be themed with textures.

## 9.315.2 Member Functions

void	<code>set_normal_texture ( Texture texture )</code>
void	<code>set_pressed_texture ( Texture texture )</code>
void	<code>set_hover_texture ( Texture texture )</code>
void	<code>set_disabled_texture ( Texture texture )</code>
void	<code>set_focused_texture ( Texture texture )</code>
void	<code>set_click_mask ( BitMap mask )</code>
void	<code>set_texture_scale ( Vector2 scale )</code>
void	<code>set_modulate ( Color color )</code>
<i>Texture</i>	<code>get_normal_texture ( ) const</code>
<i>Texture</i>	<code>get_pressed_texture ( ) const</code>
<i>Texture</i>	<code>get_hover_texture ( ) const</code>
<i>Texture</i>	<code>get_disabled_texture ( ) const</code>
<i>Texture</i>	<code>get_focused_texture ( ) const</code>
<i>BitMap</i>	<code>get_click_mask ( ) const</code>
<i>Vector2</i>	<code>get_texture_scale ( ) const</code>
<i>Color</i>	<code>get_modulate ( ) const</code>

## 9.315.3 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.

## 9.315.4 Member Function Description

- void `set_normal_texture ( Texture texture )`
- void `set_pressed_texture ( Texture texture )`
- void `set_hover_texture ( Texture texture )`
- void `set_disabled_texture ( Texture texture )`
- void `set_focused_texture ( Texture texture )`
- void `set_click_mask ( BitMap mask )`
- void `set_texture_scale ( Vector2 scale )`
- void `set_modulate ( Color color )`
- *Texture* `get_normal_texture ( ) const`
- *Texture* `get_pressed_texture ( ) const`
- *Texture* `get_hover_texture ( ) const`
- *Texture* `get_disabled_texture ( ) const`
- *Texture* `get_focused_texture ( ) const`
- *BitMap* `get_click_mask ( ) const`
- *Vector2* `get_texture_scale ( ) const`

- `Color get_modulate() const`

## 9.316 TextureFrame

**Inherits:** `Control < CanvasItem < Node < Object`

**Category:** Core

### 9.316.1 Brief Description

Control Frame that draws a texture.

### 9.316.2 Member Functions

<code>void</code>	<code>set_texture ( Object texture )</code>
<code>Object</code>	<code>get_texture () const</code>
<code>void</code>	<code>set_modulate ( Color modulate )</code>
<code>Color</code>	<code>get_modulate () const</code>
<code>void</code>	<code>set_expand ( bool enable )</code>
<code>bool</code>	<code>has_expand () const</code>

### 9.316.3 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.

### 9.316.4 Member Function Description

- `void set_texture ( Object texture )`
- `Object get_texture () const`
- `void set_modulate ( Color modulate )`
- `Color get_modulate () const`
- `void set_expand ( bool enable )`
- `bool has_expand () const`

## 9.317 TextureProgress

**Inherits:** `Range < Control < CanvasItem < Node < Object`

**Category:** Core

### 9.317.1 Brief Description

Textured progress bar implementation.

## 9.317.2 Member Functions

void	<code>set_under_texture ( Object tex )</code>
<i>Object</i>	<code>get_under_texture () const</code>
void	<code>set_progress_texture ( Object tex )</code>
<i>Object</i>	<code>get_progress_texture () const</code>
void	<code>set_over_texture ( Object tex )</code>
<i>Object</i>	<code>get_over_texture () const</code>
void	<code>set_fill_mode ( int mode )</code>
<i>int</i>	<code>get_fill_mode ()</code>
void	<code>set_radial_initial_angle ( float mode )</code>
<i>float</i>	<code>get_radial_initial_angle ()</code>
void	<code>set_radial_center_offset ( Vector2 mode )</code>
<i>Vector2</i>	<code>get_radial_center_offset ()</code>
void	<code>set_fill_degrees ( float mode )</code>
<i>float</i>	<code>get_fill_degrees ()</code>

## 9.317.3 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**

## 9.317.4 Description

*ProgressBar* implementation that is easier to theme (by just passing a few textures).

## 9.317.5 Member Function Description

- void `set_under_texture ( Object tex )`
- *Object* `get_under_texture () const`
- void `set_progress_texture ( Object tex )`
- *Object* `get_progress_texture () const`
- void `set_over_texture ( Object tex )`
- *Object* `get_over_texture () const`
- void `set_fill_mode ( int mode )`
- *int* `get_fill_mode ()`
- void `set_radial_initial_angle ( float mode )`
- *float* `get_radial_initial_angle ()`
- void `set_radial_center_offset ( Vector2 mode )`

- `Vector2 get_radial_center_offset()`
- `void set_fill_degrees (float mode)`
- `float get_fill_degrees()`

## 9.318 Theme

Inherits: [Resource](#) < [Reference](#) < [Object](#)

Category: Core

### 9.318.1 Brief Description

Theme for controls.

### 9.318.2 Member Functions

<code>void</code>	<code>set_icon ( <i>String</i> name, <i>String</i> type, <i>Texture</i> texture )</code>
<code>Texture</code>	<code>get_icon ( <i>String</i> name, <i>String</i> type ) const</code>
<code>bool</code>	<code>has_icon ( <i>String</i> name, <i>String</i> type ) const</code>
<code>void</code>	<code>clear_icon ( <i>String</i> name, <i>String</i> type )</code>
<code>StringArray</code>	<code>get_icon_list ( <i>String</i> type ) const</code>
<code>void</code>	<code>set_stylebox ( <i>String</i> name, <i>String</i> type, <i>StyleBox</i> texture )</code>
<code>StyleBox</code>	<code>get_stylebox ( <i>String</i> name, <i>String</i> type ) const</code>
<code>bool</code>	<code>has_stylebox ( <i>String</i> name, <i>String</i> type ) const</code>
<code>void</code>	<code>clear_stylebox ( <i>String</i> name, <i>String</i> type )</code>
<code>StringArray</code>	<code>get_stylebox_list ( <i>String</i> type ) const</code>
<code>void</code>	<code>set_font ( <i>String</i> name, <i>String</i> type, <i>Font</i> font )</code>
<code>Font</code>	<code>get_font ( <i>String</i> name, <i>String</i> type ) const</code>
<code>bool</code>	<code>has_font ( <i>String</i> name, <i>String</i> type ) const</code>
<code>void</code>	<code>clear_font ( <i>String</i> name, <i>String</i> type )</code>
<code>StringArray</code>	<code>get_font_list ( <i>String</i> type ) const</code>
<code>void</code>	<code>set_color ( <i>String</i> name, <i>String</i> type, <i>Color</i> color )</code>
<code>Color</code>	<code>get_color ( <i>String</i> name, <i>String</i> type ) const</code>
<code>bool</code>	<code>has_color ( <i>String</i> name, <i>String</i> type ) const</code>
<code>void</code>	<code>clear_color ( <i>String</i> name, <i>String</i> type )</code>
<code>StringArray</code>	<code>get_color_list ( <i>String</i> type ) const</code>
<code>void</code>	<code>set_constant ( <i>String</i> name, <i>String</i> type, <i>int</i> constant )</code>
<code>int</code>	<code>get_constant ( <i>String</i> name, <i>String</i> type ) const</code>
<code>bool</code>	<code>has_constant ( <i>String</i> name, <i>String</i> type ) const</code>
<code>void</code>	<code>clear_constant ( <i>String</i> name, <i>String</i> type )</code>
<code>StringArray</code>	<code>get_constant_list ( <i>String</i> type ) const</code>
<code>void</code>	<code>set_default_font ( <i>Object</i> font )</code>
<code>Object</code>	<code>get_default_font () const</code>
<code>StringArray</code>	<code>get_type_list ( <i>String</i> type ) const</code>
<code>void</code>	<code>copy_default_theme ()</code>

### 9.318.3 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 [wiki](#) for more info.

### 9.318.4 Member Function Description

- void **set\_icon** ( *String* name, *String* type, *Texture* texture )
- *Texture* **get\_icon** ( *String* name, *String* type ) const
- *bool* **has\_icon** ( *String* name, *String* type ) const
- void **clear\_icon** ( *String* name, *String* type )
- *StringArray* **get\_icon\_list** ( *String* type ) const
- void **set\_stylebox** ( *String* name, *String* type, *StyleBox* texture )
- *StyleBox* **get\_stylebox** ( *String* name, *String* type ) const
- *bool* **has\_stylebox** ( *String* name, *String* type ) const
- void **clear\_stylebox** ( *String* name, *String* type )
- *StringArray* **get\_stylebox\_list** ( *String* type ) const
- void **set\_font** ( *String* name, *String* type, *Font* font )
- *Font* **get\_font** ( *String* name, *String* type ) const
- *bool* **has\_font** ( *String* name, *String* type ) const
- void **clear\_font** ( *String* name, *String* type )
- *StringArray* **get\_font\_list** ( *String* type ) const
- void **set\_color** ( *String* name, *String* type, *Color* color )
- *Color* **get\_color** ( *String* name, *String* type ) const
- *bool* **has\_color** ( *String* name, *String* type ) const
- void **clear\_color** ( *String* name, *String* type )
- *StringArray* **get\_color\_list** ( *String* type ) const
- void **set\_constant** ( *String* name, *String* type, *int* constant )
- *int* **get\_constant** ( *String* name, *String* type ) const
- *bool* **has\_constant** ( *String* name, *String* type ) const
- void **clear\_constant** ( *String* name, *String* type )
- *StringArray* **get\_constant\_list** ( *String* type ) const
- void **set\_default\_font** ( *Object* font )
- *Object* **get\_default\_font** ( ) const
- *StringArray* **get\_type\_list** ( *String* type ) const
- void **copy\_default\_theme** ( )

## 9.319 Thread

**Inherits:** [Reference](#) < [Object](#)

**Category:** Core

### 9.319.1 Brief Description

### 9.319.2 Member Functions

Error	<code>start ( Object instance, String method, var userdata=NULL, int priority=1 )</code>
<i>String</i>	<code>get_id () const</code>
<i>bool</i>	<code>is_active () const</code>
Variant	<code>wait_to_finish ()</code>

### 9.319.3 Numeric Constants

- **PRIORITY\_LOW = 0**
- **PRIORITY\_NORMAL = 1**
- **PRIORITY\_HIGH = 2**

### 9.319.4 Member Function Description

- Error `start ( Object instance, String method, var userdata=NULL, int priority=1 )`
- *String* `get_id () const`
- *bool* `is_active () const`
- Variant `wait_to_finish ()`

## 9.320 TileMap

**Inherits:** [Node2D](#) < [CanvasItem](#) < [Node](#) < [Object](#)

**Category:** Core

### 9.320.1 Brief Description

Node for 2D tile-based games.

### 9.320.2 Member Functions

void	<code>set_tileset ( TileSet tileset )</code>
<i>TileSet</i>	<code>get_tileset () const</code>
void	<code>set_mode ( int mode )</code>
Continued on next page	

Table 9.31 – continued from previous page

<i>int</i>	<i>get_mode () const</i>
<i>void</i>	<i>set_half_offset ( int half_offset )</i>
<i>int</i>	<i>get_half_offset () const</i>
<i>void</i>	<i>set_custom_transform ( Matrix32 custom_transform )</i>
<i>Matrix32</i>	<i>get_custom_transform () const</i>
<i>void</i>	<i>set_cell_size ( Vector2 size )</i>
<i>Vector2</i>	<i>get_cell_size () const</i>
<i>void</i>	<i>set_quadrant_size ( int size )</i>
<i>int</i>	<i>get_quadrant_size () const</i>
<i>void</i>	<i>set_tile_origin ( int origin )</i>
<i>int</i>	<i>get_tile_origin () const</i>
<i>void</i>	<i>set_center_x ( bool enable )</i>
<i>bool</i>	<i>get_center_x () const</i>
<i>void</i>	<i>set_center_y ( bool enable )</i>
<i>bool</i>	<i>get_center_y () const</i>
<i>void</i>	<i>set_y_sort_mode ( bool enable )</i>
<i>bool</i>	<i>is_y_sort_mode_enabled () const</i>
<i>void</i>	<i>set_collision_use_kinematic ( bool use_kinematic )</i>
<i>bool</i>	<i>get_collision_use_kinematic () const</i>
<i>void</i>	<i>set_collision_layer ( int mask )</i>
<i>int</i>	<i>get_collision_layer () const</i>
<i>void</i>	<i>set_collision_mask ( int mask )</i>
<i>int</i>	<i>get_collision_mask () const</i>
<i>void</i>	<i>set_collision_friction ( float value )</i>
<i>float</i>	<i>get_collision_friction () const</i>
<i>void</i>	<i>set_collision_bounce ( float value )</i>
<i>float</i>	<i>get_collision_bounce () const</i>
<i>void</i>	<i>set_occluder_light_mask ( int mask )</i>
<i>int</i>	<i>get_occluder_light_mask () const</i>
<i>void</i>	<i>set_cell ( int x, int y, int tile, bool flip_x=false, bool flip_y=false, bool transpose=false )</i>
<i>void</i>	<i>set_cellv ( Vector2 pos, int tile, bool flip_x=false, bool flip_y=false, bool transpose=false )</i>
<i>int</i>	<i>get_cell ( int x, int y ) const</i>
<i>int</i>	<i>get_cellv ( Vector2 pos ) const</i>
<i>bool</i>	<i>is_cell_x_flipped ( int x, int y ) const</i>
<i>bool</i>	<i>is_cell_y_flipped ( int x, int y ) const</i>
<i>void</i>	<i>clear ()</i>
<i>Array</i>	<i>get_used_cells () const</i>
<i>Vector2</i>	<i>map_to_world ( Vector2 mappos, bool ignore_half_ofs=false ) const</i>
<i>Vector2</i>	<i>world_to_map ( Vector2 worldpos ) const</i>

### 9.320.3 Signals

- `settings_changed ()`

### 9.320.4 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.

## 9.320.5 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.

## 9.320.6 Member Function Description

- void **set\_tileset** ( *TileSet* tileset )

Set the current tileset.

- *TileSet* **get\_tileset** ( ) const

Return the current tileset.

- void **set\_mode** ( *int* mode )

Set the orientation mode as square, isometric or custom (use MODE\_\* constants as argument).

- *int* **get\_mode** ( ) const

Return the 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.

- *int* **get\_half\_offset** ( ) const

Return the current half offset configuration.

- void **set\_custom\_transform** ( *Matrix32* custom\_transform )

Set custom transform matrix, to use in combination with the custom orientation mode.

- *Matrix32* **get\_custom\_transform** ( ) const

Return the custom transform matrix.

- void **set\_cell\_size** ( *Vector2* size )

Set the cell size.

- *Vector2* **get\_cell\_size** ( ) const

Return the cell size.

- 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.

- `int get_quadrant_size () const`

Return the quadrant size.

- `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).

- `int get_tile_origin () const`

Return the tile origin configuration.

- `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).

- `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).

- `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).

- `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).

- `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.

- `bool is_y_sort_mode_enabled () const`

Return the Y sort mode.

- `void set_collision_use_kinematic ( bool use_kinematic )`

Set the tilemap to handle collisions as a kinematic body (enabled) or a static body (disabled).

- `bool get_collision_use_kinematic () const`

Return whether the tilemap handles collisions as a kinematic body.

- `void set_collision_layer ( int mask )`

Set the collision layer.

Layers are referenced by binary indexes, so allowable values to describe the 20 available layers range from 0 to  $2^{20}-1$ .

- `int get_collision_layer () const`

Return the collision layer.

- `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  $2^{20}-1$ .

- `int get_collision_mask() const`

Return the collision mask.

- `void set_collision_friction(float value)`

Set the collision friction parameter.

Allowable values range from 0 to 1.

- `float get_collision_friction() const`

Return the collision friction parameter.

- `void set_collision_bounce(float value)`

Set the collision bounce parameter.

Allowable values range from 0 to 1.

- `float get_collision_bounce() const`

Return the collision bounce parameter.

- `void set_occluder_light_mask(int mask)`

- `int get_occluder_light_mask() const`

- `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_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.

- `int get_cell(int x, int y) const`

Return the tile index of the referenced cell.

- `int get_cellv(Vector2 pos) const`

Return the tile index of the cell referenced by a Vector2.

- `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.

- `void clear()`

Clear all cells.

- `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).

- `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.

- `Vector2 world_to_map ( Vector2 worldpos ) const`

Return the tilemap (grid-based) coordinates corresponding to the absolute world position given as an argument.

## 9.321 TileSet

**Inherits:** `Resource < Reference < Object`

**Category:** Core

### 9.321.1 Brief Description

Tile library for tilemaps.

### 9.321.2 Member Functions

void	<code>create_tile ( int id )</code>
void	<code>tile_set_name ( int id, String name )</code>
<i>String</i>	<code>tile_get_name ( int id ) const</code>
void	<code>tile_set_texture ( int id, Texture texture )</code>
<i>Texture</i>	<code>tile_get_texture ( int id ) const</code>
void	<code>tile_set_material ( int id, CanvasItemMaterial material )</code>
<i>CanvasItemMaterial</i>	<code>tile_get_material ( int id ) const</code>
void	<code>tile_set_texture_offset ( int id, Vector2 texture_offset )</code>
<i>Vector2</i>	<code>tile_get_texture_offset ( int id ) const</code>
void	<code>tile_set_shape_offset ( int id, Vector2 shape_offset )</code>
<i>Vector2</i>	<code>tile_get_shape_offset ( int id ) const</code>
void	<code>tile_set_region ( int id, Rect2 region )</code>
<i>Rect2</i>	<code>tile_get_region ( int id ) const</code>
void	<code>tile_set_shape ( int id, Shape2D shape )</code>
<i>Shape2D</i>	<code>tile_get_shape ( int id ) const</code>
void	<code>tile_set_shapes ( int id, Array shapes )</code>
<i>Array</i>	<code>tile_get_shapes ( int id ) const</code>
void	<code>tile_set_navigation_polygon ( int id, NavigationPolygon navigation_polygon )</code>
<i>NavigationPolygon</i>	<code>tile_get_navigation_polygon ( int id ) const</code>
void	<code>tile_set_navigation_polygon_offset ( int id, Vector2 navigation_polygon_offset )</code>
<i>Vector2</i>	<code>tile_get_navigation_polygon_offset ( int id ) const</code>
void	<code>tile_set_light_occluder ( int id, OccluderPolygon2D light_occluder )</code>
<i>OccluderPolygon2D</i>	<code>tile_get_light_occluder ( int id ) const</code>
void	<code>tile_set_occluder_offset ( int id, Vector2 occluder_offset )</code>
<i>Vector2</i>	<code>tile_get_occluder_offset ( int id ) const</code>
void	<code>remove_tile ( int id )</code>
void	<code>clear ()</code>
<i>int</i>	<code>get_last_unused_tile_id () const</code>
<i>int</i>	<code>find_tile_by_name ( String name ) const</code>
<i>Array</i>	<code>get_tiles_ids () const</code>

### 9.321.3 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.

### 9.321.4 Member Function Description

- void `create_tile ( int id )`

Create a new tile which will be referenced by the given ID.

- void `tile_set_name ( int id, String name )`

Set the name of the tile, for descriptive purposes.

- *String* `tile_get_name ( int id ) const`

Return the name of the tile.

- `void tile_set_texture ( int id, Texture texture )`

Set the texture of the tile.

- `Texture tile_get_texture ( int id ) const`

Return the texture of the tile.

- `void tile_set_material ( int id, CanvasItemMaterial material )`

Set the material of the tile.

- `CanvasItemMaterial tile_get_material ( int id ) const`

Return the material of the tile.

- `void tile_set_texture_offset ( int id, Vector2 texture_offset )`

Set the texture offset of the tile.

- `Vector2 tile_get_texture_offset ( int id ) const`

Return the texture offset of the tile.

- `void tile_set_shape_offset ( int id, Vector2 shape_offset )`

Set the shape offset of the tile.

- `Vector2 tile_get_shape_offset ( int id ) const`

Return the shape offset of the tile.

- `void tile_set_region ( int id, Rect2 region )`

Set the tile sub-region in the texture. This is common in texture atlases.

- `Rect2 tile_get_region ( int id ) const`

Return the tile sub-region in the texture.

- `void tile_set_shape ( int id, Shape2D shape )`

Set a shape for the tile, enabling physics to collide with it.

- `Shape2D tile_get_shape ( int id ) const`

Return the shape 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.

- `Array tile_get_shapes ( int id ) const`

Return the array of shapes of the tile.

- `void tile_set_navigation_polygon ( int id, NavigationPolygon navigation_polygon )`

Set a navigation polygon for the tile.

- `NavigationPolygon tile_get_navigation_polygon ( int id ) const`

Return the navigation polygon of the tile.

- `void tile_set_navigation_polygon_offset ( int id, Vector2 navigation_polygon_offset )`

Set an offset for the tile's navigation polygon.

- `Vector2 tile_get_navigation_polygon_offset ( int id ) const`

Return the offset of the tile's navigation polygon.

- void **tile\_set\_light\_occluder** ( *int* id, *OccluderPolygon2D* light\_occluder )

Set a light occluder for the tile.

- *OccluderPolygon2D* **tile\_get\_light\_occluder** ( *int* id ) const

Return the light occluder of the tile.

- void **tile\_set\_occluder\_offset** ( *int* id, *Vector2* occluder\_offset )

Set an offset for the tile's light occluder.

- *Vector2* **tile\_get\_occluder\_offset** ( *int* id ) const

Return the offset of the tile's light occluder.

- void **remove\_tile** ( *int* id )

Remove the tile referenced by the given ID.

- void **clear** ( )

Clear all tiles.

- *int* **get\_last\_unused\_tile\_id** ( ) const

Return the ID following the last currently used ID, useful when creating a new tile.

- *int* **find\_tile\_by\_name** ( *String* name ) const

Find the first tile matching the given name.

- *Array* **get\_tiles\_ids** ( ) const

Return an array of all currently used tile IDs.

## 9.322 Timer

**Inherits:** *Node < Object*

**Category:** Core

### 9.322.1 Brief Description

### 9.322.2 Member Functions

void	<i>set_wait_time</i> ( <i>float</i> time_sec )
<i>float</i>	<i>get_wait_time</i> ( ) const
void	<i>set_one_shot</i> ( <i>bool</i> enable )
<i>bool</i>	<i>is_one_shot</i> ( ) const
void	<i>set_autostart</i> ( <i>bool</i> enable )
<i>bool</i>	<i>has_autostart</i> ( ) const
void	<i>start</i> ( )
void	<i>stop</i> ( )
<i>float</i>	<i>get_time_left</i> ( ) const
void	<i>set_timer_process_mode</i> ( <i>int</i> mode )
<i>int</i>	<i>get_timer_process_mode</i> ( ) const

### 9.322.3 Signals

- `timeout()`

### 9.322.4 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.

### 9.322.5 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.

### 9.322.6 Member Function Description

- `void set_wait_time (float time_sec)`

Set wait time in seconds. When the time is over, it will emit the timeout signal.

- `float get_wait_time () const`

Return the wait time in seconds.

- `void set_one_shot (bool enable)`

Set as one-shot. If enabled, the timer will stop after timeout, otherwise it will automatically restart.

- `bool is_one_shot () const`

Return true if configured as one-shot.

- `void set_autostart (bool enable)`

Set to automatically start when entering the scene.

- `bool has_autostart () const`

Return true if set to automatically start when entering the scene.

- `void start ()`

Start the timer.

- `void stop ()`

Stop (cancel) the timer.

- `float get_time_left () const`

Return the time left for timeout in seconds if the timer is active, 0 otherwise.

- `void set_timer_process_mode (int mode)`

Set the timer's processing mode (fixed or idle, use `TIMER_PROCESS_*` constants as argument).

- `int get_timer_process_mode () const`

Return the timer's processing mode.

## 9.323 ToolButton

**Inherits:** `Button < BaseButton < Control < CanvasItem < Node < Object`

**Category:** Core

### 9.323.1 Brief Description

## 9.324 TouchScreenButton

**Inherits:** `Node2D < CanvasItem < Node < Object`

**Category:** Core

### 9.324.1 Brief Description

### 9.324.2 Member Functions

void	<code>set_texture ( Object texture )</code>
<i>Object</i>	<code>get_texture () const</code>
void	<code>set_texture_pressed ( Object texture_pressed )</code>
<i>Object</i>	<code>get_texture_pressed () const</code>
void	<code>set_bitmask ( Object bitmask )</code>
<i>Object</i>	<code>get_bitmask () const</code>
void	<code>set_action ( String action )</code>
<i>String</i>	<code>get_action () const</code>
void	<code>set_visibility_mode ( int mode )</code>
<i>int</i>	<code>get_visibility_mode () const</code>
void	<code>set_passby_press ( bool enabled )</code>
<i>bool</i>	<code>is_passby_press_enabled () const</code>
<i>bool</i>	<code>is_pressed () const</code>

### 9.324.3 Signals

- `released ()`
- `pressed ()`

### 9.324.4 Member Function Description

- void `set_texture ( Object texture )`
- *Object* `get_texture () const`
- void `set_texture_pressed ( Object texture_pressed )`
- *Object* `get_texture_pressed () const`
- void `set_bitmask ( Object bitmask )`
- *Object* `get_bitmask () const`
- void `set_action ( String action )`

- `String get_action () const`
- `void set_visibility_mode ( int mode )`
- `int get_visibility_mode () const`
- `void set_passby_press ( bool enabled )`
- `bool is_passby_press_enabled () const`
- `bool is_pressed () const`

## 9.325 Transform

**Category:** Built-In Types

### 9.325.1 Brief Description

3D Transformation.

### 9.325.2 Member Functions

<code>Transform</code>	<code>affine_inverse ()</code>
<code>Transform</code>	<code>inverse ()</code>
<code>Transform</code>	<code>looking_at ( Vector3 target, Vector3 up )</code>
<code>Transform</code>	<code>orthonormalized ()</code>
<code>Transform</code>	<code>rotated ( Vector3 axis, float phi )</code>
<code>Transform</code>	<code>scaled ( Vector3 scale )</code>
<code>Transform</code>	<code>translated ( Vector3 ofs )</code>
<code>var</code>	<code>xform ( var v )</code>
<code>var</code>	<code>xform_inv ( var v )</code>
<code>Transform</code>	<code>Transform ( Vector3 x_axis, Vector3 y_axis, Vector3 z_axis, Vector3 origin )</code>
<code>Transform</code>	<code>Transform ( Matrix3 basis, Vector3 origin )</code>
<code>Transform</code>	<code>Transform ( Matrix32 from )</code>
<code>Transform</code>	<code>Transform ( Quat from )</code>
<code>Transform</code>	<code>Transform ( Matrix3 from )</code>

### 9.325.3 Member Variables

- `Matrix3 basis`
- `Vector3 origin`

### 9.325.4 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.

## 9.325.5 Member Function Description

- *Transform* **affine\_inverse ()**
- *Transform* **inverse ()**

Returns the inverse of the transform.

- *Transform* **looking\_at ( Vector3 target, Vector3 up )**
- *Transform* **orthonormalized ()**
- *Transform* **rotated ( Vector3 axis, float phi )**
- *Transform* **scaled ( Vector3 scale )**
- *Transform* **translated ( Vector3 ofs )**
- var **xform ( var v )**

Transforms vector “v” by this transform.

- var **xform\_inv ( var v )**

Inverse-transforms vector “v” by this transform.

- *Transform* **Transform ( Vector3 x\_axis, Vector3 y\_axis, Vector3 z\_axis, Vector3 origin )**
- *Transform* **Transform ( Matrix3 basis, Vector3 origin )**
- *Transform* **Transform ( Matrix32 from )**
- *Transform* **Transform ( Quat from )**
- *Transform* **Transform ( Matrix3 from )**

## 9.326 Translation

**Inherits:** *Resource < Reference < Object*

**Inherited By:** *PHashTranslation*

**Category:** Core

### 9.326.1 Brief Description

Language Translation.

### 9.326.2 Member Functions

void	<i>set_locale ( String locale )</i>
<i>String</i>	<i>get_locale () const</i>
void	<i>add_message ( String src_message, String xlated_message )</i>
<i>String</i>	<i>get_message ( String src_message ) const</i>
void	<i>erase_message ( String src_message )</i>
<i>StringArray</i>	<i>get_message_list () const</i>
<i>int</i>	<i>get_message_count () const</i>

### 9.326.3 Description

Translations are resources that can be loaded/unloaded on demand. They map a string to another string.

### 9.326.4 Member Function Description

- void **set\_locale** ( *String* locale )

Set the locale of the translation.

- *String* **get\_locale** ( ) const

Return the locale of the translation.

- void **add\_message** ( *String* src\_message, *String* xlated\_message )

Add a message for translation.

- *String* **get\_message** ( *String* src\_message ) const

Return a message for translation.

- void **erase\_message** ( *String* src\_message )

Erase a message.

- *StringArray* **get\_message\_list** ( ) const

Return all the messages (keys).

- *int* **get\_message\_count** ( ) const

## 9.327 TranslationServer

**Inherits:** *Object*

**Category:** Core

### 9.327.1 Brief Description

Server that manages all translations. Translations can be set to it and removed from it.

### 9.327.2 Member Functions

void	<i>set_locale</i> ( <i>String</i> locale )
<i>String</i>	<i>get_locale</i> ( ) const
<i>String</i>	<i>translate</i> ( <i>String</i> message ) const
void	<i>add_translation</i> ( <i>Translation</i> translation )
void	<i>remove_translation</i> ( <i>Translation</i> translation )
void	<i>clear</i> ( )

### 9.327.3 Member Function Description

- void `set_locale` ( `String` locale )
- `String get_locale` ( ) const
- `String translate` ( `String` message ) const
- void `add_translation` ( `Translation` translation )
- void `remove_translation` ( `Translation` translation )
- void `clear` ( )

## 9.328 Tree

**Inherits:** `Control < CanvasItem < Node < Object`

**Category:** Core

### 9.328.1 Brief Description

### 9.328.2 Member Functions

void	<code>clear</code> ( )
<code>TreeItem</code>	<code>create_item</code> ( <code>TreeItem</code> parent=Object() )
<code>TreeItem</code>	<code>get_root</code> ( )
void	<code>set_column_min_width</code> ( <code>int</code> column, <code>int</code> min_width )
void	<code>set_column_expand</code> ( <code>int</code> column, <code>bool</code> expand )
<code>int</code>	<code>get_column_width</code> ( <code>int</code> column ) const
void	<code>set_hide_root</code> ( <code>bool</code> enable )
<code>TreeItem</code>	<code>get_next_selected</code> ( <code>TreeItem</code> from )
<code>TreeItem</code>	<code>get_selected</code> ( ) const
<code>int</code>	<code>get_selected_column</code> ( ) const
<code>int</code>	<code>get_pressed_button</code> ( ) const
void	<code>set_select_mode</code> ( <code>int</code> mode )
void	<code>set_columns</code> ( <code>int</code> amount )
<code>int</code>	<code>get_columns</code> ( ) const
<code>TreeItem</code>	<code>get_edited</code> ( ) const
<code>int</code>	<code>get_edited_column</code> ( ) const
<code>Rect2</code>	<code>get_custom_popup_rect</code> ( ) const
<code>Rect2</code>	<code>get_item_area_rect</code> ( <code>TreeItem</code> item, <code>int</code> column=-1 ) const
void	<code>ensure_cursor_is_visible</code> ( )
void	<code>set_column_titles_visible</code> ( <code>bool</code> visible )
<code>bool</code>	<code>are_column_titles_visible</code> ( ) const
void	<code>set_column_title</code> ( <code>int</code> column, <code>String</code> title )
<code>String</code>	<code>get_column_title</code> ( <code>int</code> column ) const
<code>Vector2</code>	<code>get_scroll</code> ( ) const
void	<code>set_hide_folding</code> ( <code>bool</code> hide )
<code>bool</code>	<code>is_folding_hidden</code> ( ) const

### 9.328.3 Signals

- **item\_activated ()**
- **multi\_selected ( *Object* item, *int* column, *bool* selected )**
- **custom\_popup\_edited ( *bool* arrow\_clicked )**
- **item\_collapsed ( *Object* item )**
- **item\_edited ()**
- **item\_selected ()**
- **cell\_selected ()**
- **button\_pressed ( *Object* item, *int* column, *int* id )**

### 9.328.4 Numeric Constants

- **SELECT\_SINGLE = 0**
- **SELECT\_ROW = 1**
- **SELECT\_MULTI = 2**

### 9.328.5 Member Function Description

- void **clear ()**
- *TreeItem* **create\_item ( *TreeItem* parent=Object() )**
- *TreeItem* **get\_root ()**
- void **set\_column\_min\_width ( *int* column, *int* min\_width )**
- void **set\_column\_expand ( *int* column, *bool* expand )**
- *int* **get\_column\_width ( *int* column ) const**
- void **set\_hide\_root ( *bool* enable )**
- *TreeItem* **get\_next\_selected ( *TreeItem* from )**
- *TreeItem* **get\_selected () const**
- *int* **get\_selected\_column () const**
- *int* **get\_pressed\_button () const**
- void **set\_select\_mode ( *int* mode )**
- void **set\_columns ( *int* amount )**
- *int* **get\_columns () const**
- *TreeItem* **get\_edited () const**
- *int* **get\_edited\_column () const**
- *Rect2* **get\_custom\_popup\_rect () const**
- *Rect2* **get\_item\_area\_rect ( *TreeItem* item, *int* column=-1 ) const**
- void **ensure\_cursor\_is\_visible ()**

- void `set_column_titles_visible` ( `bool` visible )
- `bool` `are_column_titles_visible` ( ) const
- void `set_column_title` ( `int` column, `String` title )
- `String` `get_column_title` ( `int` column ) const
- `Vector2` `get_scroll` ( ) const
- void `set_hide_folding` ( `bool` hide )
- `bool` `is_folding_hidden` ( ) const

## 9.329 TreeItem

**Inherits:** `Object`

**Category:** Core

### 9.329.1 Brief Description

### 9.329.2 Member Functions

<code>void</code>	<code>set_cell_mode</code> ( <code>int</code> column, <code>int</code> mode )
<code>int</code>	<code>get_cell_mode</code> ( <code>int</code> column ) const
<code>void</code>	<code>set_checked</code> ( <code>int</code> column, <code>bool</code> checked )
<code>bool</code>	<code>is_checked</code> ( <code>int</code> column ) const
<code>void</code>	<code>set_text</code> ( <code>int</code> column, <code>String</code> text )
<code>String</code>	<code>get_text</code> ( <code>int</code> column ) const
<code>void</code>	<code>set_icon</code> ( <code>int</code> column, <code>Texture</code> texture )
<code>Texture</code>	<code>get_icon</code> ( <code>int</code> column ) const
<code>void</code>	<code>set_icon_region</code> ( <code>int</code> column, <code>Rect2</code> region )
<code>Rect2</code>	<code>get_icon_region</code> ( <code>int</code> column ) const
<code>void</code>	<code>set_icon_max_width</code> ( <code>int</code> column, <code>int</code> width )
<code>int</code>	<code>get_icon_max_width</code> ( <code>int</code> column ) const
<code>void</code>	<code>set_range</code> ( <code>int</code> column, <code>float</code> value )
<code>float</code>	<code>get_range</code> ( <code>int</code> column ) const
<code>void</code>	<code>set_range_config</code> ( <code>int</code> column, <code>float</code> min, <code>float</code> max, <code>float</code> step, <code>bool</code> expr=false )
<code>Dictionary</code>	<code>get_range_config</code> ( <code>int</code> column )
<code>void</code>	<code>set_metadata</code> ( <code>int</code> column, var meta )
<code>void</code>	<code>get_metadata</code> ( <code>int</code> column ) const
<code>void</code>	<code>set_custom_draw</code> ( <code>int</code> column, <code>Object</code> object, <code>String</code> callback )
<code>void</code>	<code>set_collapsed</code> ( <code>bool</code> enable )
<code>bool</code>	<code>is_collapsed</code> ( )
<code>TreeItem</code>	<code>get_next</code> ( )
<code>TreeItem</code>	<code>get_prev</code> ( )
<code>TreeItem</code>	<code>get_parent</code> ( )
<code>TreeItem</code>	<code>get_children</code> ( )
<code>TreeItem</code>	<code>get_next_visible</code> ( )
<code>TreeItem</code>	<code>get_prev_visible</code> ( )
<code>TreeItem</code>	<code>remove_child</code> ( <code>Object</code> child )

Continued on next page

Table 9.32 – continued from previous page

void	<code>set_selectable( int column, bool selectable )</code>
<i>bool</i>	<code>is_selectable( int column ) const</code>
<i>bool</i>	<code>is_selected( int column )</code>
void	<code>select( int column )</code>
void	<code>deselect( int column )</code>
void	<code>set_editable( int column, bool enabled )</code>
<i>bool</i>	<code>is_editable( int column )</code>
void	<code>set_custom_color( int column, Color color )</code>
void	<code>clear_custom_color( int column )</code>
void	<code>set_custom_bg_color( int column, Color color )</code>
void	<code>clear_custom_bg_color( int column )</code>
<i>Color</i>	<code>get_custom_bg_color( int column ) const</code>
void	<code>add_button( int column, Texture button, int button_idx=-1, bool disabled=false )</code>
<i>int</i>	<code>get_button_count( int column ) const</code>
<i>Texture</i>	<code>get_button( int column, int button_idx ) const</code>
void	<code>erase_button( int column, int button_idx )</code>
<i>bool</i>	<code>is_button_disabled( int column, int button_idx ) const</code>
void	<code>set_tooltip( int column, String tooltip )</code>
<i>String</i>	<code>get_tooltip( int column ) const</code>
void	<code>move_to_top()</code>
void	<code>move_to_bottom()</code>

### 9.329.3 Numeric Constants

- **CELL\_MODE\_STRING = 0**
- **CELL\_MODE\_CHECK = 1**
- **CELL\_MODE\_RANGE = 2**
- **CELL\_MODE\_ICON = 3**
- **CELL\_MODE\_CUSTOM = 4**

### 9.329.4 Member Function Description

- void `set_cell_mode( int column, int mode )`
- *int* `get_cell_mode( int column ) const`
- void `set_checked( int column, bool checked )`
- *bool* `is_checked( int column ) const`
- void `set_text( int column, String text )`
- *String* `get_text( int column ) const`
- void `set_icon( int column, Texture texture )`
- *Texture* `get_icon( int column ) const`
- void `set_icon_region( int column, Rect2 region )`
- *Rect2* `get_icon_region( int column ) const`
- void `set_icon_max_width( int column, int width )`

- `int get_icon_max_width ( int column ) const`
- `void set_range ( int column, float value )`
- `float get_range ( int column ) const`
- `void set_range_config ( int column, float min, float max, float step, bool expr=false )`
- `Dictionary get_range_config ( int column )`
- `void set_metadata ( int column, var meta )`
- `void get_metadata ( int column ) const`
- `void set_custom_draw ( int column, Object object, String callback )`
- `void set_collapsed ( bool enable )`
- `bool is_collapsed ()`
- `TreeItem get_next ()`
- `TreeItem get_prev ()`
- `TreeItem get_parent ()`
- `TreeItem get_children ()`
- `TreeItem get_next_visible ()`
- `TreeItem get_prev_visible ()`
- `TreeItem remove_child ( Object child )`
- `void set_selectable ( int column, bool selectable )`
- `bool is_selectable ( int column ) const`
- `bool is_selected ( int column )`
- `void select ( int column )`
- `void deselect ( int column )`
- `void set_editable ( int column, bool enabled )`
- `bool is_editable ( int column )`
- `void set_custom_color ( int column, Color color )`
- `void clear_custom_color ( int column )`
- `void set_custom_bg_color ( int column, Color color )`
- `void clear_custom_bg_color ( int column )`
- `Color get_custom_bg_color ( int column ) const`
- `void add_button ( int column, Texture button, int button_idx=-1, bool disabled=false )`
- `int get_button_count ( int column ) const`
- `Texture get_button ( int column, int button_idx ) const`
- `void erase_button ( int column, int button_idx )`
- `bool is_button_disabled ( int column, int button_idx ) const`
- `void set_tooltip ( int column, String tooltip )`
- `String get_tooltip ( int column ) const`

- void **move\_to\_top ()**
- void **move\_to\_bottom ()**

## 9.330 Tween

**Inherits:** *Node < Object*

**Category:** Core

### 9.330.1 Brief Description

### 9.330.2 Member Functions

<code>bool</code>	<code>is_active () const</code>
<code>void</code>	<code>set_active ( bool active )</code>
<code>bool</code>	<code>is_repeat () const</code>
<code>void</code>	<code>set_repeat ( bool repeat )</code>
<code>void</code>	<code>set_speed ( float speed )</code>
<code>float</code>	<code>get_speed () const</code>
<code>void</code>	<code>set_tween_process_mode ( int mode )</code>
<code>int</code>	<code>get_tween_process_mode () const</code>
<code>bool</code>	<code>start ()</code>
<code>bool</code>	<code>reset ( Object object, String key )</code>
<code>bool</code>	<code>reset_all ()</code>
<code>bool</code>	<code>stop ( Object object, String key )</code>
<code>bool</code>	<code>stop_all ()</code>
<code>bool</code>	<code>resume ( Object object, String key )</code>
<code>bool</code>	<code>resume_all ()</code>
<code>bool</code>	<code>remove ( Object object, String key )</code>
<code>bool</code>	<code>remove_all ()</code>
<code>bool</code>	<code>seek ( float time )</code>
<code>float</code>	<code>tell () const</code>
<code>float</code>	<code>get_runtime () const</code>
<code>bool</code>	<code>interpolate_property ( Object object, String property, var initial_val, var final_val, float times_in_sec, int trans_type, int ease_type, float delay=0 )</code>
<code>bool</code>	<code>interpolate_method ( Object object, String method, var initial_val, var final_val, float times_in_sec, int trans_type, int ease_type, float delay=0 )</code>
<code>bool</code>	<code>interpolate_callback ( Object object, float times_in_sec, String callback, var arg1=NULL, var arg2=NULL, var arg3=NULL, var arg4=NULL, var arg5=NULL )</code>
<code>bool</code>	<code>interpolate_deferred_callback ( Object object, float times_in_sec, String callback, var arg1=NULL, var arg2=NULL, var arg3=NULL, var arg4=NULL, var arg5=NULL )</code>
<code>bool</code>	<code>follow_property ( Object object, String property, var initial_val, Object target, String target_property, float times_in_sec, int trans_type, int ease_type, float delay=0 )</code>
<code>bool</code>	<code>follow_method ( Object object, String method, var initial_val, Object target, String target_method, float times_in_sec, int trans_type, int ease_type, float delay=0 )</code>
<code>bool</code>	<code>targeting_property ( Object object, String property, Object initial, String initial_val, var final_val, float times_in_sec, int trans_type, int ease_type, float delay=0 )</code>
<code>bool</code>	<code>targeting_method ( Object object, String method, Object initial, String initial_method, var final_val, float times_in_sec, int trans_type, int ease_type, float delay=0 )</code>

### 9.330.3 Signals

- `tween_complete ( Object object, String key )`
- `tween_step ( Object object, String key, float elapsed, Object value )`
- `tween_start ( Object object, String key )`

### 9.330.4 Numeric Constants

- `TWEEN_PROCESS_FIXED = 0`
- `TWEEN_PROCESS_IDLE = 1`
- `TRANS_LINEAR = 0`
- `TRANS_SINE = 1`
- `TRANS_QUINT = 2`
- `TRANS_QUART = 3`
- `TRANS_QUAD = 4`
- `TRANS_EXPO = 5`
- `TRANS_ELASTIC = 6`
- `TRANS_CUBIC = 7`
- `TRANS_CIRC = 8`
- `TRANS_BOUNCE = 9`
- `TRANS_BACK = 10`
- `EASE_IN = 0`
- `EASE_OUT = 1`
- `EASE_IN_OUT = 2`
- `EASE_OUT_IN = 3`

### 9.330.5 Member Function Description

- `bool is_active () const`
- `void set_active ( bool active )`
- `bool is_repeat () const`
- `void set_repeat ( bool repeat )`
- `void set_speed ( float speed )`
- `float get_speed () const`
- `void set_tween_process_mode ( int mode )`
- `int get_tween_process_mode () const`
- `bool start ()`
- `bool reset ( Object object, String key )`

- `bool reset_all ()`
- `bool stop ( Object object, String key )`
- `bool stop_all ()`
- `bool resume ( Object object, String key )`
- `bool resume_all ()`
- `bool remove ( Object object, String key )`
- `bool remove_all ()`
- `bool seek ( float time )`
- `float tell () const`
- `float get_runtime () const`
- `bool interpolate_property ( Object object, String property, var initial_val, var final_val, float times_in_sec, int trans_type, int ease_type, float delay=0 )`
- `bool interpolate_method ( Object object, String method, var initial_val, var final_val, float times_in_sec, int trans_type, int ease_type, float delay=0 )`
- `bool interpolate_callback ( Object object, float times_in_sec, String callback, var arg1=NULL, var arg2=NULL, var arg3=NULL, var arg4=NULL, var arg5=NULL )`
- `bool interpolate_deferred_callback ( Object object, float times_in_sec, String callback, var arg1=NULL, var arg2=NULL, var arg3=NULL, var arg4=NULL, var arg5=NULL )`
- `bool follow_property ( Object object, String property, var initial_val, Object target, String target_property, float times_in_sec, int trans_type, int ease_type, float delay=0 )`
- `bool follow_method ( Object object, String method, var initial_val, Object target, String target_method, 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, var final_val, float times_in_sec, int trans_type, int ease_type, float delay=0 )`
- `bool targeting_method ( Object object, String method, Object initial, String initial_method, var final_val, float times_in_sec, int trans_type, int ease_type, float delay=0 )`

## 9.331 UndoRedo

Inherits: `Object`

Category: Core

### 9.331.1 Brief Description

### 9.331.2 Member Functions

void	<code>create_action ( String name, bool mergeable=false )</code>
void	<code>commit_action ()</code>
void	<code>add_do_method ( Object object, String method, var arg0=NULL, var arg1=NULL, var arg2=NULL, var arg3=NULL, var arg4=NULL )</code>
void	<code>add_undo_method ( Object object, String method, var arg0=NULL, var arg1=NULL, var arg2=NULL, var arg3=NULL, var arg4=NULL )</code>
void	<code>add_do_property ( Object object, String property, Variant value )</code>
void	<code>add_undo_property ( Object object, String property, Variant value )</code>
void	<code>add_do_reference ( Object object )</code>
void	<code>add_undo_reference ( Object object )</code>
void	<code>clear_history ()</code>
<code>String</code>	<code>get_current_action_name () const</code>
<code>int</code>	<code>get_version () const</code>

### 9.331.3 Member Function Description

- void `create_action ( String name, bool mergeable=false )`
- void `commit_action ()`
- void `add_do_method ( Object object, String method, var arg0=NULL, var arg1=NULL, var arg2=NULL, var arg3=NULL, var arg4=NULL )`
- void `add_undo_method ( Object object, String method, var arg0=NULL, var arg1=NULL, var arg2=NULL, var arg3=NULL, var arg4=NULL )`
- void `add_do_property ( Object object, String property, Variant value )`
- void `add_undo_property ( Object object, String property, Variant value )`
- void `add_do_reference ( Object object )`
- void `add_undo_reference ( Object object )`
- void `clear_history ()`
- `String get_current_action_name () const`
- `int get_version () const`

## 9.332 VBoxContainer

**Inherits:** `BoxContainer < Container < Control < CanvasItem < Node < Object`

**Category:** Core

### 9.332.1 Brief Description

Vertical box container.

### 9.332.2 Description

Vertical box container. See [BoxContainer](#).

## 9.333 VButtonArray

**Inherits:** [ButtonArray](#) < [Control](#) < [CanvasItem](#) < [Node](#) < [Object](#)

**Category:** Core

### 9.333.1 Brief Description

Vertical button array.

### 9.333.2 Description

Vertical button array. See [ButtonArray](#).

## 9.334 Vector2

**Category:** Built-In Types

### 9.334.1 Brief Description

Vector used for 2D Math.

## 9.334.2 Member Functions

<i>float</i>	<code>angle ()</code>
<i>float</i>	<code>angle_to ( Vector2 to )</code>
<i>float</i>	<code>angle_to_point ( Vector2 to )</code>
<i>Vector2</i>	<code>cubic_interpolate ( Vector2 b, Vector2 pre_a, Vector2 post_b, float t )</code>
<i>float</i>	<code>distance_squared_to ( Vector2 to )</code>
<i>float</i>	<code>distance_to ( Vector2 to )</code>
<i>float</i>	<code>dot ( Vector2 with )</code>
<i>Vector2</i>	<code>floor ()</code>
<i>Vector2</i>	<code>floorf ()</code>
<i>float</i>	<code>get_aspect ()</code>
<i>float</i>	<code>length ()</code>
<i>float</i>	<code>length_squared ()</code>
<i>Vector2</i>	<code>linear_interpolate ( Vector2 b, float t )</code>
<i>Vector2</i>	<code>normalized ()</code>
<i>Vector2</i>	<code>reflect ( Vector2 vec )</code>
<i>Vector2</i>	<code>rotated ( float phi )</code>
<i>Vector2</i>	<code>slide ( Vector2 vec )</code>
<i>Vector2</i>	<code>snapped ( Vector2 by )</code>
<i>Vector2</i>	<code>tangent ()</code>
<i>Vector2</i>	<code>Vector2 ( float x, float y )</code>

## 9.334.3 Member Variables

- *float* **x**
- *float* **y**
- *float* **width**
- *float* **height**

## 9.334.4 Member Function Description

- *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* **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** ( )

- *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.

- *Vector2* **Vector2** ( *float* x, *float* y )

Constructs a new Vector2 from the given x and y.

## 9.335 Vector2Array

**Category:** Built-In Types

### 9.335.1 Brief Description

An Array of Vector2.

### 9.335.2 Member Functions

<code>void</code>	<code>push_back ( Vector2 vector2 )</code>
<code>void</code>	<code>resize ( int idx )</code>
<code>void</code>	<code>set ( int idx, Vector2 vector2 )</code>
<code>int</code>	<code>size ( )</code>
<code>Vector2Array</code>	<code>Vector2Array ( Array from )</code>

### 9.335.3 Description

An Array specifically designed to hold Vector2.

### 9.335.4 Member Function Description

- `void push_back ( Vector2 vector2 )`

Inserts a Vector2 at the end.

- `void resize ( int idx )`

Sets 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 )`

Changes the Vector2 at the given index.

- `int size ( )`

Returns the size of the array.

- `Vector2Array Vector2Array ( Array from )`

Constructs a new Vector2Array. Optionally, you can pass in an Array that will be converted.

## 9.336 Vector3

**Category:** Built-In Types

### 9.336.1 Brief Description

Vector class, which performs basic 3D vector math operations.

## 9.336.2 Member Functions

<i>Vector3</i>	<code>abs ()</code>
<i>Vector3</i>	<code>ceil ()</code>
<i>Vector3</i>	<code>cross ( Vector3 b )</code>
<i>Vector3</i>	<code>cubic_interpolate ( Vector3 b, Vector3 pre_a, Vector3 post_b, float t )</code>
<i>float</i>	<code>distance_squared_to ( Vector3 b )</code>
<i>float</i>	<code>distance_to ( Vector3 b )</code>
<i>float</i>	<code>dot ( Vector3 b )</code>
<i>Vector3</i>	<code>floor ()</code>
<i>Vector3</i>	<code>inverse ()</code>
<i>float</i>	<code>length ()</code>
<i>float</i>	<code>length_squared ()</code>
<i>Vector3</i>	<code>linear_interpolate ( Vector3 b, float t )</code>
<i>int</i>	<code>max_axis ()</code>
<i>int</i>	<code>min_axis ()</code>
<i>Vector3</i>	<code>normalized ()</code>
<i>Vector3</i>	<code>reflect ( Vector3 by )</code>
<i>Vector3</i>	<code>rotated ( Vector3 axis, float phi )</code>
<i>Vector3</i>	<code>slide ( Vector3 by )</code>
<i>Vector3</i>	<code>snapped ( float by )</code>
<i>Vector3</i>	<code>Vector3 ( float x, float y, float z )</code>

## 9.336.3 Member Variables

- *float x*
- *float y*
- *float z*

## 9.336.4 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.

## 9.336.5 Description

`Vector3` is one of the core classes of the engine, and includes several built-in helper functions to perform basic vector math operations.

## 9.336.6 Member Function Description

- *Vector3 abs ()*

Returns a new vector with all components in absolute values (e.g. positive).

- *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.

- `Vector3 Vector3 ( float x, float y, float z )`

Returns a Vector3 with the given components.

## 9.337 Vector3Array

**Category:** Built-In Types

### 9.337.1 Brief Description

An Array of Vector3.

### 9.337.2 Member Functions

void	<code>push_back ( Vector3 vector3 )</code>
void	<code>resize ( int idx )</code>
void	<code>set ( int idx, Vector3 vector3 )</code>
int	<code>size ()</code>
<code>Vector3Array</code>	<code>Vector3Array ( Array from )</code>

### 9.337.3 Description

An Array specifically designed to hold Vector3.

### 9.337.4 Member Function Description

- void `push_back ( Vector3 vector3 )`

Inserts a Vector3 at the end.

- void `resize ( int idx )`

Sets 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 )`

Changes the Vector3 at the given index.

- `int size ()`

Returns the size of the array.

- `Vector3Array Vector3Array ( Array from )`

Constructs a new Vector3Array. Optionally, you can pass in an Array that will be converted.

## 9.338 VehicleBody

**Inherits:** `PhysicsBody < CollisionObject < Spatial < Node < Object`

**Category:** Core

### 9.338.1 Brief Description

### 9.338.2 Member Functions

void	<code>set_mass (float mass )</code>
<code>float</code>	<code>get_mass () const</code>
void	<code>set_friction (float friction )</code>
<code>float</code>	<code>get_friction () const</code>
void	<code>set_engine_force (float engine_force )</code>
<code>float</code>	<code>get_engine_force () const</code>
void	<code>set_brake (float brake )</code>
<code>float</code>	<code>get_brake () const</code>
void	<code>set_steering (float steering )</code>
<code>float</code>	<code>get_steering () const</code>

### 9.338.3 Member Function Description

- void `set_mass (float mass )`
- `float get_mass () const`
- void `set_friction (float friction )`
- `float get_friction () const`
- void `set_engine_force (float engine_force )`
- `float get_engine_force () const`
- void `set_brake (float brake )`
- `float get_brake () const`
- void `set_steering (float steering )`
- `float get_steering () const`

## 9.339 VehicleWheel

Inherits: *Spatial < Node < Object*

Category: Core

### 9.339.1 Brief Description

### 9.339.2 Member Functions

void	<code>set_radius (float length)</code>
<code>float</code>	<code>get_radius () const</code>
void	<code>set_suspension_rest_length (float length)</code>
<code>float</code>	<code>get_suspension_rest_length () const</code>
void	<code>set_suspension_travel (float length)</code>
<code>float</code>	<code>get_suspension_travel () const</code>
void	<code>set_suspension_stiffness (float length)</code>
<code>float</code>	<code>get_suspension_stiffness () const</code>
void	<code>set_suspension_max_force (float length)</code>
<code>float</code>	<code>get_suspension_max_force () const</code>
void	<code>set_damping_compression (float length)</code>
<code>float</code>	<code>get_damping_compression () const</code>
void	<code>set_damping_relaxation (float length)</code>
<code>float</code>	<code>get_damping_relaxation () const</code>
void	<code>set_use_as_traction (bool enable)</code>
<code>bool</code>	<code>is_used_as_traction () const</code>
void	<code>set_use_as_steering (bool enable)</code>
<code>bool</code>	<code>is_used_as_steering () const</code>
void	<code>set_friction_slip (float length)</code>
<code>float</code>	<code>get_friction_slip () const</code>

### 9.339.3 Member Function Description

- void `set_radius (float length)`
- `float get_radius () const`
- void `set_suspension_rest_length (float length)`
- `float get_suspension_rest_length () const`
- void `set_suspension_travel (float length)`
- `float get_suspension_travel () const`
- void `set_suspension_stiffness (float length)`
- `float get_suspension_stiffness () const`
- void `set_suspension_max_force (float length)`
- `float get_suspension_max_force () const`
- void `set_damping_compression (float length)`
- `float get_damping_compression () const`
- void `set_damping_relaxation (float length)`
- `float get_damping_relaxation () const`
- void `set_use_as_traction (bool enable)`
- `bool is_used_as_traction () const`
- void `set_use_as_steering (bool enable)`

- `bool is_used_as_steering() const`
- `void set_friction_slip (float length)`
- `float get_friction_slip () const`

## 9.340 VideoPlayer

**Inherits:** `Control` < `CanvasItem` < `Node` < `Object`

**Category:** Core

### 9.340.1 Brief Description

### 9.340.2 Member Functions

<code>void</code>	<code>set_stream ( Stream stream )</code>
<code>Stream</code>	<code>get_stream () const</code>
<code>void</code>	<code>play ()</code>
<code>void</code>	<code>stop ()</code>
<code>bool</code>	<code>is_playing () const</code>
<code>void</code>	<code>set_paused (<code>bool</code> paused)</code>
<code>bool</code>	<code>is_paused () const</code>
<code>void</code>	<code>set_volume (<code>float</code> volume)</code>
<code>float</code>	<code>get_volume () const</code>
<code>void</code>	<code>set_volume_db (<code>float</code> db)</code>
<code>float</code>	<code>get_volume_db () const</code>
<code>void</code>	<code>set_audio_track (<code>int</code> track)</code>
<code>int</code>	<code>get_audio_track () const</code>
<code>String</code>	<code>get_stream_name () const</code>
<code>float</code>	<code>get_stream_pos () const</code>
<code>void</code>	<code>set_autoplay (<code>bool</code> enabled)</code>
<code>bool</code>	<code>has_autoplay () const</code>
<code>void</code>	<code>set_expand (<code>bool</code> enable)</code>
<code>bool</code>	<code>has_expand () const</code>
<code>void</code>	<code>set_buffering_msec (<code>int</code> msec)</code>
<code>int</code>	<code>get_buffering_msec () const</code>
<code>Texture</code>	<code>get_video_texture ()</code>

### 9.340.3 Member Function Description

- `void set_stream ( Stream stream )`
- `Stream get_stream () const`
- `void play ()`
- `void stop ()`
- `bool is_playing () const`
- `void set_paused (bool paused)`
- `bool is_paused () const`

- void **set\_volume** (*float* volume )
- *float* **get\_volume** () const
- void **set\_volume\_db** (*float* db )
- *float* **get\_volume\_db** () const
- void **set\_audio\_track** (*int* track )
- *int* **get\_audio\_track** () const
- *String* **get\_stream\_name** () const
- *float* **get\_stream\_pos** () const
- void **set\_autoplay** (*bool* enabled )
- *bool* **has\_autoplay** () const
- void **set\_expand** (*bool* enable )
- *bool* **has\_expand** () const
- void **set\_buffering\_msec** (*int* msec )
- *int* **get\_buffering\_msec** () const
- *Texture* **get\_video\_texture** ()

## 9.341 VideoStream

**Inherits:** *Resource* < *Reference* < *Object*

**Inherited By:** *VideoStreamTheora*

**Category:** Core

### 9.341.1 Brief Description

## 9.342 VideoStreamTheora

**Inherits:** *VideoStream* < *Resource* < *Reference* < *Object*

**Category:** Core

### 9.342.1 Brief Description

## 9.343 Viewport

**Inherits:** *Node* < *Object*

**Category:** Core

### 9.343.1 Brief Description

Creates a sub-view into the screen.

## 9.343.2 Member Functions

void	<code>set_rect ( Rect2 rect )</code>
<i>Rect2</i>	<code>get_rect ( ) const</code>
<i>World2D</i>	<code>find_world_2d ( ) const</code>
void	<code>set_world ( World world )</code>
<i>World</i>	<code>get_world ( ) const</code>
<i>World</i>	<code>find_world ( ) const</code>
void	<code>set_canvas_transform ( Matrix32 xform )</code>
<i>Matrix32</i>	<code>get_canvas_transform ( ) const</code>
void	<code>set_global_canvas_transform ( Matrix32 xform )</code>
<i>Matrix32</i>	<code>get_global_canvas_transform ( ) const</code>
<i>Matrix32</i>	<code>get_final_transform ( ) const</code>
<i>Rect2</i>	<code>get_visible_rect ( ) const</code>
void	<code>set_transparent_background ( bool enable )</code>
<i>bool</i>	<code>has_transparent_background ( ) const</code>
void	<code>set_size_override ( bool enable, Vector2 size=Vector2(-1,-1), Vector2 margin=Vector2(0,0) )</code>
<i>Vector2</i>	<code>get_size_override ( ) const</code>
<i>bool</i>	<code>is_size_override_enabled ( ) const</code>
void	<code>set_size_override_stretch ( bool enabled )</code>
<i>bool</i>	<code>is_size_override_stretch_enabled ( ) const</code>
void	<code>queue_screen_capture ( )</code>
<i>Image</i>	<code>get_screen_capture ( ) const</code>
void	<code>set_as_render_target ( bool enable )</code>
<i>bool</i>	<code>is_set_as_render_target ( ) const</code>
void	<code>set_render_target_vflip ( bool enable )</code>
<i>bool</i>	<code>get_render_target_vflip ( ) const</code>
void	<code>set_render_target_clear_on_new_frame ( bool enable )</code>
<i>bool</i>	<code>get_render_target_clear_on_new_frame ( ) const</code>
void	<code>render_target_clear ( )</code>
void	<code>set_render_target_filter ( bool enable )</code>
<i>bool</i>	<code>get_render_target_filter ( ) const</code>
void	<code>set_render_target_gen_mipmaps ( bool enable )</code>
<i>bool</i>	<code>get_render_target_gen_mipmaps ( ) const</code>
void	<code>set_render_target_update_mode ( int mode )</code>
<i>int</i>	<code>get_render_target_update_mode ( ) const</code>
<i>RenderTargetTexture</i>	<code>get_render_target_texture ( ) const</code>
void	<code>set_physics_object_picking ( bool enable )</code>
<i>bool</i>	<code>get_physics_object_picking ( )</code>
<i>RID</i>	<code>get_viewport ( ) const</code>
void	<code>input ( InputEvent local_event )</code>
void	<code>unhandled_input ( InputEvent local_event )</code>
void	<code>update_worlds ( )</code>
void	<code>set_use_own_world ( bool enable )</code>
<i>bool</i>	<code>is_using_own_world ( ) const</code>
<i>Camera</i>	<code>get_camera ( ) const</code>
void	<code>set_as_audio_listener ( bool enable )</code>
<i>bool</i>	<code>is_audio_listener ( ) const</code>
void	<code>set_as_audio_listener_2d ( bool enable )</code>
<i>bool</i>	<code>is_audio_listener_2d ( ) const</code>
void	<code>set_render_target_to_screen_rect ( Rect2 rect )</code>

Continued on next page

Table 9.33 – continued from previous page

<i>Vector2</i>	<code>get_mouse_pos () const</code>
<code>void</code>	<code>warp_mouse ( <i>Vector2</i> to_pos )</code>
<code>bool</code>	<code>gui_has_modal_stack () const</code>
<code>void</code>	<code>set_disable_input ( <code>bool</code> disable )</code>
<code>bool</code>	<code>is_input_disabled () const</code>

### 9.343.3 Signals

- `size_changed ()`

### 9.343.4 Numeric Constants

- `RENDER_TARGET_UPDATE_DISABLED = 0`
- `RENDER_TARGET_UPDATE_ONCE = 1`
- `RENDER_TARGET_UPDATE_WHEN_VISIBLE = 2`
- `RENDER_TARGET_UPDATE_ALWAYS = 3`

### 9.343.5 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.

### 9.343.6 Member Function Description

- `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.

- `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.

- `World2D find_world_2d () const`
- `void set_world ( World world )`
- `World get_world () const`
- `World find_world () const`

- void **set\_canvas\_transform** (*Matrix32* xform)
- *Matrix32* **get\_canvas\_transform** () const
- void **set\_global\_canvas\_transform** (*Matrix32* xform)
- *Matrix32* **get\_global\_canvas\_transform** () const
- *Matrix32* **get\_final\_transform** () const
- *Rect2* **get\_visible\_rect** () const

Return the final, visible rect in global screen coordinates.

- void **set\_transparent\_background** (*bool* enable)

If this viewport is a child of another viewport, keep the previously drawn background visible.

- *bool* **has\_transparent\_background** () const

Return whether the viewport lets whatever is behind it to show.

- void **set\_size\_override** (*bool* enable, *Vector2* size=Vector2(-1,-1), *Vector2* margin=Vector2(0,0))
- *Vector2* **get\_size\_override** () const
- *bool* **is\_size\_override\_enabled** () const
- void **set\_size\_override\_stretch** (*bool* enabled)
- *bool* **is\_size\_override\_stretch\_enabled** () const
- void **queue\_screen\_capture** ()
- *Image* **get\_screen\_capture** () const
- void **set\_as\_render\_target** (*bool* enable)
- *bool* **is\_set\_as\_render\_target** () const
- void **set\_render\_target\_vflip** (*bool* enable)
- *bool* **get\_render\_target\_vflip** () const
- void **set\_render\_target\_clear\_on\_new\_frame** (*bool* enable)
- *bool* **get\_render\_target\_clear\_on\_new\_frame** () const
- void **render\_target\_clear** ()
- void **set\_render\_target\_filter** (*bool* enable)
- *bool* **get\_render\_target\_filter** () const
- void **set\_render\_target\_gen\_mipmaps** (*bool* enable)
- *bool* **get\_render\_target\_gen\_mipmaps** () const
- void **set\_render\_target\_update\_mode** (*int* mode)
- *int* **get\_render\_target\_update\_mode** () const
- *RenderTargetTexture* **get\_render\_target\_texture** () const
- void **set\_physics\_object\_picking** (*bool* enable)
- *bool* **get\_physics\_object\_picking** ()
- *RID* **get\_viewport** () const

Get the viewport RID from the visual server.

- void **input** ( *InputEvent* local\_event )
- void **unhandled\_input** ( *InputEvent* local\_event )
- void **update\_worlds** ()
- void **set\_use\_own\_world** ( *bool* enable )
- *bool* **is\_using\_own\_world** () const
- *Camera* **get\_camera** () const
- void **set\_as\_audio\_listener** ( *bool* enable )
- *bool* **is\_audio\_listener** () const
- void **set\_as\_audio\_listener\_2d** ( *bool* enable )
- *bool* **is\_audio\_listener\_2d** () const
- void **set\_render\_target\_to\_screen\_rect** ( *Rect2* rect )
- *Vector2* **get\_mouse\_pos** () const
- void **warp\_mouse** ( *Vector2* to\_pos )
- *bool* **gui\_has\_modal\_stack** () const
- void **set\_disable\_input** ( *bool* disable )
- *bool* **is\_input\_disabled** () const

## 9.344 ViewportSprite

Inherits: *Node2D* < *CanvasItem* < *Node* < *Object*

Category: Core

### 9.344.1 Brief Description

### 9.344.2 Member Functions

void	<i>set_viewport_path</i> ( <i>NodePath</i> path )
<i>NodePath</i>	<i>get_viewport_path</i> () const
void	<i>set_centered</i> ( <i>bool</i> centered )
<i>bool</i>	<i>is_centered</i> () const
void	<i>set_offset</i> ( <i>Vector2</i> offset )
<i>Vector2</i>	<i>get_offset</i> () const
void	<i>set_modulate</i> ( <i>Color</i> modulate )
<i>Color</i>	<i>get_modulate</i> () const

### 9.344.3 Member Function Description

- void **set\_viewport\_path** ( *NodePath* path )
- *NodePath* **get\_viewport\_path** () const
- void **set\_centered** ( *bool* centered )

- `bool is_centered () const`
- `void set_offset ( Vector2 offset )`
- `Vector2 get_offset () const`
- `void set_modulate ( Color modulate )`
- `Color get_modulate () const`

## 9.345 VisibilityEnabler

**Inherits:** `VisibilityNotifier < Spatial < Node < Object`

**Category:** Core

### 9.345.1 Brief Description

### 9.345.2 Member Functions

<code>void</code>	<code>set_enabler ( int enabler, bool enabled )</code>
<code>bool</code>	<code>is_enabler_enabled ( int enabler ) const</code>

### 9.345.3 Numeric Constants

- `ENABLER_FREEZE_BODIES = 1`
- `ENABLER_PAUSE_ANIMATIONS = 0`
- `ENABLER_MAX = 2`

### 9.345.4 Member Function Description

- `void set_enabler ( int enabler, bool enabled )`
- `bool is_enabler_enabled ( int enabler ) const`

## 9.346 VisibilityEnabler2D

**Inherits:** `VisibilityNotifier2D < Node2D < CanvasItem < Node < Object`

**Category:** Core

### 9.346.1 Brief Description

### 9.346.2 Member Functions

<code>void</code>	<code>set_enabler ( int enabler, bool enabled )</code>
<code>bool</code>	<code>is_enabler_enabled ( int enabler ) const</code>

### 9.346.3 Numeric Constants

- **ENABLER\_FREEZE\_BODIES** = 1
- **ENABLER\_PAUSE\_ANIMATIONS** = 0
- **ENABLER\_PAUSE\_PARTICLES** = 2
- **ENABLER\_PARENT\_PROCESS** = 3
- **ENABLER\_PARENT\_FIXED\_PROCESS** = 4
- **ENABLER\_MAX** = 5

### 9.346.4 Member Function Description

- void **set\_enabler** ( *int* enabler, *bool* enabled )
- *bool* **is\_enabler\_enabled** ( *int* enabler ) const

## 9.347 VisibilityNotifier

**Inherits:** *Spatial* < *Node* < *Object*

**Inherited By:** *VisibilityEnabler*

**Category:** Core

### 9.347.1 Brief Description

### 9.347.2 Member Functions

<i>void</i>	<b>set_aabb</b> ( <i>AABB</i> rect )
<i>AABB</i>	<b>get_aabb</b> ( ) const
<i>bool</i>	<b>is_on_screen</b> ( ) const

### 9.347.3 Signals

- **enter\_screen** ( )
- **enter\_camera** ( *Object* camera )
- **exit\_screen** ( )
- **exit\_camera** ( *Object* camera )

### 9.347.4 Member Function Description

- void **set\_aabb** ( *AABB* rect )
- *AABB* **get\_aabb** ( ) const
- *bool* **is\_on\_screen** ( ) const

## 9.348 VisibilityNotifier2D

**Inherits:** [Node2D](#) < [CanvasItem](#) < [Node](#) < [Object](#)

**Inherited By:** [VisibilityEnabler2D](#)

**Category:** Core

### 9.348.1 Brief Description

### 9.348.2 Member Functions

void	<code>set_rect ( Rect2 rect )</code>
<i>Rect2</i>	<code>get_rect ( ) const</code>
<i>bool</i>	<code>is_on_screen ( ) const</code>

### 9.348.3 Signals

- `enter_screen ( )`
- `enter_viewport ( Object viewport )`
- `exit_screen ( )`
- `exit_viewport ( Object viewport )`

### 9.348.4 Member Function Description

- void `set_rect ( Rect2 rect )`
- *Rect2* `get_rect ( ) const`
- *bool* `is_on_screen ( ) const`

## 9.349 VisualInstance

**Inherits:** [Spatial](#) < [Node](#) < [Object](#)

**Inherited By:** [BakedLightInstance](#), [Light](#), [Room](#), [BakedLightSampler](#), [Portal](#), [GeometryInstance](#)

**Category:** Core

### 9.349.1 Brief Description

### 9.349.2 Member Functions

void	<code>set_base ( RID base )</code>
void	<code>set_layer_mask ( int mask )</code>
<i>int</i>	<code>get_layer_mask ( ) const</code>

### 9.349.3 Member Function Description

- void `set_base ( RID base )`
- void `set_layer_mask ( int mask )`
- `int get_layer_mask ( ) const`

## 9.350 VisualServer

**Inherits:** *Object*

**Category:** Core

### 9.350.1 Brief Description

Server for anything visible.

### 9.350.2 Member Functions

<i>RID</i>	<code>texture_create ()</code>
<i>RID</i>	<code>texture_create_from_image ( Image arg0, int arg1=7 )</code>
void	<code>texture_set_flags ( RID arg0, int arg1 )</code>
<i>int</i>	<code>texture_get_flags ( RID arg0 ) const</code>
<i>int</i>	<code>texture_get_width ( RID arg0 ) const</code>
<i>int</i>	<code>texture_get_height ( RID arg0 ) const</code>
<i>RID</i>	<code>shader_create ( int mode=0 )</code>
void	<code>shader_set_mode ( RID shader, int mode )</code>
<i>RID</i>	<code>material_create ()</code>
void	<code>material_set_shader ( RID shader, RID arg1 )</code>
<i>RID</i>	<code>material_get_shader ( RID arg0 ) const</code>
void	<code>material_set_param ( RID arg0, String arg1, var arg2 )</code>
void	<code>material_get_param ( RID arg0, String arg1 ) const</code>
void	<code>material_set_flag ( RID arg0, int arg1, bool arg2 )</code>
<i>bool</i>	<code>material_get_flag ( RID arg0, int arg1 ) const</code>
void	<code>material_set_blend_mode ( RID arg0, int arg1 )</code>
<i>int</i>	<code>material_get_blend_mode ( RID arg0 ) const</code>
void	<code>material_set_line_width ( RID arg0, float arg1 )</code>
<i>float</i>	<code>material_get_line_width ( RID arg0 ) const</code>
<i>RID</i>	<code>mesh_create ()</code>
void	<code>mesh_add_surface ( RID arg0, int arg1, Array arg2, Array arg3, bool arg4=-1 )</code>
void	<code>mesh_surface_set_material ( RID arg0, int arg1, RID arg2, bool arg3=false )</code>
<i>RID</i>	<code>mesh_surface_get_material ( RID arg0, int arg1 ) const</code>
<i>int</i>	<code>mesh_surface_get_array_len ( RID arg0, int arg1 ) const</code>
<i>int</i>	<code>mesh_surface_get_array_index_len ( RID arg0, int arg1 ) const</code>
<i>int</i>	<code>mesh_surface_get_format ( RID arg0, int arg1 ) const</code>
<i>int</i>	<code>mesh_surface_get_primitive_type ( RID arg0, int arg1 ) const</code>
void	<code>mesh_remove_surface ( RID arg0, int arg1 )</code>
<i>int</i>	<code>mesh_get_surface_count ( RID arg0 ) const</code>

Continued on

Table 9.34 – continued from previous page

<i>RID</i>	<i>multimesh_create ()</i>
void	<i>multimesh_set_mesh ( RID arg0, RID arg1 )</i>
void	<i>multimesh_set_aabb ( RID arg0, AABB arg1 )</i>
void	<i>multimesh_instance_set_transform ( RID arg0, int arg1, Transform arg2 )</i>
void	<i>multimesh_instance_set_color ( RID arg0, int arg1, Color arg2 )</i>
<i>RID</i>	<i>multimesh_get_mesh ( RID arg0 ) const</i>
<i>AABB</i>	<i>multimesh_get_aabb ( RID arg0, AABB arg1 ) const</i>
<i>Transform</i>	<i>multimesh_instance_get_transform ( RID arg0, int arg1 ) const</i>
<i>Color</i>	<i>multimesh_instance_get_color ( RID arg0, int arg1 ) const</i>
<i>RID</i>	<i>particles_create ()</i>
void	<i>particles_set_amount ( RID arg0, int arg1 )</i>
<i>int</i>	<i>particles_get_amount ( RID arg0 ) const</i>
void	<i>particles_set_emitting ( RID arg0, bool arg1 )</i>
<i>bool</i>	<i>particles_is_emitting ( RID arg0 ) const</i>
void	<i>particles_set_visibility_aabb ( RID arg0, AABB arg1 )</i>
<i>AABB</i>	<i>particles_get_visibility_aabb ( RID arg0 ) const</i>
void	<i>particles_set_variable ( RID arg0, int arg1, float arg2 )</i>
<i>float</i>	<i>particles_get_variable ( RID arg0, int arg1 ) const</i>
void	<i>particles_set_randomness ( RID arg0, int arg1, float arg2 )</i>
<i>float</i>	<i>particles_get_randomness ( RID arg0, int arg1 ) const</i>
void	<i>particles_set_color_phases ( RID arg0, int arg1 )</i>
<i>int</i>	<i>particles_get_color_phases ( RID arg0 ) const</i>
void	<i>particles_set_color_phase_pos ( RID arg0, int arg1, float arg2 )</i>
<i>float</i>	<i>particles_get_color_phase_pos ( RID arg0, int arg1 ) const</i>
void	<i>particles_set_color_phase_color ( RID arg0, int arg1, Color arg2 )</i>
<i>Color</i>	<i>particles_get_color_phase_color ( RID arg0, int arg1 ) const</i>
void	<i>particles_set_attractors ( RID arg0, int arg1 )</i>
<i>int</i>	<i>particles_get_attractors ( RID arg0 ) const</i>
void	<i>particles_set_attractor_pos ( RID arg0, int arg1, Vector3 arg2 )</i>
<i>Vector3</i>	<i>particles_get_attractor_pos ( RID arg0, int arg1 ) const</i>
void	<i>particles_set_attractor_strength ( RID arg0, int arg1, float arg2 )</i>
<i>float</i>	<i>particles_get_attractor_strength ( RID arg0, int arg1 ) const</i>
void	<i>particles_set_material ( RID arg0, RID arg1, bool arg2=false )</i>
void	<i>particles_set_height_from_velocity ( RID arg0, bool arg1 )</i>
<i>bool</i>	<i>particles_has_height_from_velocity ( RID arg0 ) const</i>
<i>RID</i>	<i>light_create ( int arg0 )</i>
<i>int</i>	<i>light_get_type ( RID arg0 ) const</i>
void	<i>light_set_color ( RID arg0, int arg1, Color arg2 )</i>
<i>Color</i>	<i>light_get_color ( RID arg0, int arg1 ) const</i>
void	<i>light_set_shadow ( RID arg0, bool arg1 )</i>
<i>bool</i>	<i>light_has_shadow ( RID arg0 ) const</i>
void	<i>light_set_volumetric ( RID arg0, bool arg1 )</i>
<i>bool</i>	<i>light_is_volumetric ( RID arg0 ) const</i>
void	<i>light_set_projector ( RID arg0, RID arg1 )</i>
<i>RID</i>	<i>light_get_projector ( RID arg0 ) const</i>
void	<i>light_set_var ( RID arg0, int arg1, float arg2 )</i>
<i>float</i>	<i>light_get_var ( RID arg0, int arg1 ) const</i>
<i>RID</i>	<i>skeleton_create ()</i>
void	<i>skeleton_resize ( RID arg0, int arg1 )</i>
<i>int</i>	<i>skeleton_get_bone_count ( RID arg0 ) const</i>

Continued on

Table 9.34 – continued from previous page

void	<code>skeleton_bone_set_transform ( RID arg0, int arg1, Transform arg2 )</code>
<code>Transform</code>	<code>skeleton_bone_get_transform ( RID arg0, int arg1 )</code>
<code>RID</code>	<code>room_create ()</code>
void	<code>room_set_bounds ( RID arg0, Dictionary arg1 )</code>
<code>Dictionary</code>	<code>room_get_bounds ( RID arg0 ) const</code>
<code>RID</code>	<code>portal_create ()</code>
void	<code>portal_set_shape ( RID arg0, Vector2Array arg1 )</code>
<code>Vector2Array</code>	<code>portal_get_shape ( RID arg0 ) const</code>
void	<code>portal_set_enabled ( RID arg0, bool arg1 )</code>
<code>bool</code>	<code>portal_is_enabled ( RID arg0 ) const</code>
void	<code>portal_set_disable_distance ( RID arg0, float arg1 )</code>
<code>float</code>	<code>portal_get_disable_distance ( RID arg0 ) const</code>
void	<code>portal_set_disabled_color ( RID arg0, Color arg1 )</code>
<code>Color</code>	<code>portal_get_disabled_color ( RID arg0 ) const</code>
<code>RID</code>	<code>camera_create ()</code>
void	<code>camera_set_perspective ( RID arg0, float arg1, float arg2, float arg3 )</code>
void	<code>camera_set_orthogonal ( RID arg0, float arg1, float arg2, float arg3 )</code>
void	<code>camera_set_transform ( RID arg0, Transform arg1 )</code>
<code>RID</code>	<code>viewport_create ()</code>
void	<code>viewport_set_rect ( RID arg0, Rect2 arg1 )</code>
<code>Rect2</code>	<code>viewport_get_rect ( RID arg0 ) const</code>
void	<code>viewport_attach_camera ( RID arg0, RID arg1=RID() )</code>
<code>RID</code>	<code>viewport_get_attached_camera ( RID arg0 ) const</code>
<code>RID</code>	<code>viewport_get_scenario ( RID arg0 ) const</code>
void	<code>viewport_attach_canvas ( RID arg0, RID arg1 )</code>
void	<code>viewport_remove_canvas ( RID arg0, RID arg1 )</code>
void	<code>viewport_set_global_canvas_transform ( RID arg0, Matrix32 arg1 )</code>
<code>RID</code>	<code>scenario_create ()</code>
void	<code>scenario_set_debug ( RID arg0, int arg1 )</code>
<code>RID</code>	<code>instance_create ()</code>
<code>RID</code>	<code>instance_get_base ( RID arg0 ) const</code>
<code>RID</code>	<code>instance_get_base_aabb ( RID arg0 ) const</code>
void	<code>instance_set_transform ( RID arg0, Transform arg1 )</code>
<code>Transform</code>	<code>instance_get_transform ( RID arg0 ) const</code>
void	<code>instance_attach_object_instance_ID ( RID arg0, int arg1 )</code>
<code>int</code>	<code>instance_get_object_instance_ID ( RID arg0 ) const</code>
void	<code>instance_attach_skeleton ( RID arg0, RID arg1 )</code>
<code>RID</code>	<code>instance_get_skeleton ( RID arg0 ) const</code>
void	<code>instance_set_room ( RID arg0, RID arg1 )</code>
<code>RID</code>	<code>instance_get_room ( RID arg0 ) const</code>
void	<code>instance_set_exterior ( RID arg0, bool arg1 )</code>
<code>bool</code>	<code>instance_is_exterior ( RID arg0 ) const</code>
<code>Array</code>	<code>instances_cull_aabb ( AABB arg0, RID arg1 ) const</code>
<code>Array</code>	<code>instances_cull_ray ( Vector3 arg0, Vector3 arg1, RID arg2 ) const</code>
<code>Array</code>	<code>instances_cull_convex ( Array arg0, RID arg1 ) const</code>
<code>RID</code>	<code>instance_geometry_override_material_param ( RID arg0 ) const</code>
<code>RID</code>	<code>instance_geometry_get_material_param ( RID arg0 ) const</code>
<code>RID</code>	<code>get_test_cube ()</code>
<code>RID</code>	<code>canvas_create ()</code>
<code>RID</code>	<code>canvas_item_create ()</code>

Continued on

Table 9.34 – continued from previous page

void	<i>canvas_item_set_parent</i> ( <i>RID</i> arg0, <i>RID</i> arg1 )
<i>RID</i>	<i>canvas_item_get_parent</i> ( <i>RID</i> arg0 ) const
void	<i>canvas_item_set_transform</i> ( <i>RID</i> arg0, <i>Matrix32</i> arg1 )
void	<i>canvas_item_set_custom_rect</i> ( <i>RID</i> arg0, <i>bool</i> arg1, <i>Rect2</i> arg2 )
void	<i>canvas_item_set_clip</i> ( <i>RID</i> arg0, <i>bool</i> arg1 )
void	<i>canvas_item_set_opacity</i> ( <i>RID</i> arg0, <i>float</i> arg1 )
<i>float</i>	<i>canvas_item_get_opacity</i> ( <i>RID</i> arg0, <i>float</i> arg1 ) const
void	<i>canvas_item_set_self_opacity</i> ( <i>RID</i> arg0, <i>float</i> arg1 )
<i>float</i>	<i>canvas_item_get_self_opacity</i> ( <i>RID</i> arg0, <i>float</i> arg1 ) const
void	<i>canvas_item_set_z</i> ( <i>RID</i> arg0, <i>int</i> arg1 )
void	<i>canvas_item_add_line</i> ( <i>RID</i> arg0, <i>Vector2</i> arg1, <i>Vector2</i> arg2, <i>Color</i> arg3, <i>float</i> arg4=1 )
void	<i>canvas_item_add_rect</i> ( <i>RID</i> arg0, <i>Rect2</i> arg1, <i>Color</i> arg2 )
void	<i>canvas_item_add_texture_rect</i> ( <i>RID</i> arg0, <i>Rect2</i> arg1, <i>RID</i> arg2, <i>bool</i> arg3, <i>Color</i> arg4=Color(1,1,1,1), <i>bool</i> arg5=false )
void	<i>canvas_item_add_texture_rect_region</i> ( <i>RID</i> arg0, <i>Rect2</i> arg1, <i>RID</i> arg2, <i>Rect2</i> arg3, <i>Color</i> arg4=Color(1,1,1,1), <i>bool</i> arg5=false )
void	<i>canvas_item_add_style_box</i> ( <i>RID</i> arg0, <i>Rect2</i> arg1, <i>RID</i> arg2, <i>RealArray</i> arg3, <i>Color</i> arg4=Color(1,1,1,1) )
void	<i>canvas_item_add_circle</i> ( <i>RID</i> arg0, <i>Vector2</i> arg1, <i>float</i> arg2, <i>Color</i> arg3 )
void	<i>viewport_set_canvas_transform</i> ( <i>RID</i> arg0, <i>RID</i> arg1, <i>Matrix32</i> arg2 )
void	<i>canvas_item_clear</i> ( <i>RID</i> arg0 )
void	<i>canvas_item_raise</i> ( <i>RID</i> arg0 )
void	<i>cursor_set_rotation</i> ( <i>float</i> arg0, <i>int</i> arg1 )
void	<i>cursor_set_texture</i> ( <i>RID</i> arg0, <i>Vector2</i> arg1, <i>int</i> arg2 )
void	<i>cursor_set_visible</i> ( <i>bool</i> arg0, <i>int</i> arg1 )
void	<i>cursor_set_pos</i> ( <i>Vector2</i> arg0, <i>int</i> arg1 )
void	<i>black_bars_set_margins</i> ( <i>int</i> left, <i>int</i> top, <i>int</i> right, <i>int</i> bottom )
void	<i>black_bars_set_images</i> ( <i>RID</i> left, <i>RID</i> top, <i>RID</i> right, <i>RID</i> bottom )
<i>RID</i>	<i>make_sphere_mesh</i> ( <i>int</i> arg0, <i>int</i> arg1, <i>float</i> arg2 )
void	<i>mesh_add_surface_from_planes</i> ( <i>RID</i> arg0, <i>Array</i> arg1 )
void	<i>draw</i> ()
void	<i>sync</i> ()
void	<i>free</i> ( <i>RID</i> arg0 )
void	<i>set_default_clear_color</i> ( <i>Color</i> arg0 )
<i>int</i>	<i>get_render_info</i> ( <i>int</i> arg0 )

### 9.350.3 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

#### 9.350.4 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.

## 9.350.5 Member Function Description

- `RID texture_create()`
- `RID texture_create_from_image( Image arg0, int arg1=7 )`
- `void texture_set_flags( RID arg0, int arg1 )`
- `int texture_get_flags( RID arg0 ) const`
- `int texture_get_width( RID arg0 ) const`
- `int texture_get_height( RID arg0 ) const`
- `RID shader_create( int mode=0 )`
- `void shader_set_mode( RID shader, int mode )`
- `RID material_create()`
- `void material_set_shader( RID shader, RID arg1 )`
- `RID material_get_shader( RID arg0 ) const`
- `void material_set_param( RID arg0, String arg1, var arg2 )`
- `void material_get_param( RID arg0, String arg1 ) const`
- `void material_set_flag( RID arg0, int arg1, bool arg2 )`
- `bool material_get_flag( RID arg0, int arg1 ) const`
- `void material_set_blend_mode( RID arg0, int arg1 )`
- `int material_get_blend_mode( RID arg0 ) const`
- `void material_set_line_width( RID arg0, float arg1 )`
- `float material_get_line_width( RID arg0 ) const`
- `RID mesh_create()`
- `void mesh_add_surface( RID arg0, int arg1, Array arg2, Array arg3, bool arg4=-1 )`
- `void mesh_surface_set_material( RID arg0, int arg1, RID arg2, bool arg3=false )`
- `RID mesh_surface_get_material( RID arg0, int arg1 ) const`
- `int mesh_surface_get_array_len( RID arg0, int arg1 ) const`
- `int mesh_surface_get_array_index_len( RID arg0, int arg1 ) const`
- `int mesh_surface_get_format( RID arg0, int arg1 ) const`
- `int mesh_surface_get_primitive_type( RID arg0, int arg1 ) const`
- `void mesh_remove_surface( RID arg0, int arg1 )`
- `int mesh_get_surface_count( RID arg0 ) const`
- `RID multimesh_create()`
- `void multimesh_set_mesh( RID arg0, RID arg1 )`
- `void multimesh_set_aabb( RID arg0, AABB arg1 )`
- `void multimesh_instance_set_transform( RID arg0, int arg1, Transform arg2 )`
- `void multimesh_instance_set_color( RID arg0, int arg1, Color arg2 )`
- `RID multimesh_get_mesh( RID arg0 ) const`

- `AABB multimesh_get_aabb ( RID arg0, AABB arg1 ) const`
- `Transform multimesh_instance_get_transform ( RID arg0, int arg1 ) const`
- `Color multimesh_instance_get_color ( RID arg0, int arg1 ) const`
- `RID particles_create ( )`
- `void particles_set_amount ( RID arg0, int arg1 )`
- `int particles_get_amount ( RID arg0 ) const`
- `void particles_set_emitting ( RID arg0, bool arg1 )`
- `bool particles_is_emitting ( RID arg0 ) const`
- `void particles_set_visibility_aabb ( RID arg0, AABB arg1 )`
- `AABB particles_get_visibility_aabb ( RID arg0 ) const`
- `void particles_set_variable ( RID arg0, int arg1, float arg2 )`
- `float particles_get_variable ( RID arg0, int arg1 ) const`
- `void particles_set_randomness ( RID arg0, int arg1, float arg2 )`
- `float particles_get_randomness ( RID arg0, int arg1 ) const`
- `void particles_set_color_phases ( RID arg0, int arg1 )`
- `int particles_get_color_phases ( RID arg0 ) const`
- `void particles_set_color_phase_pos ( RID arg0, int arg1, float arg2 )`
- `float particles_get_color_phase_pos ( RID arg0, int arg1 ) const`
- `void particles_set_color_phase_color ( RID arg0, int arg1, Color arg2 )`
- `Color particles_get_color_phase_color ( RID arg0, int arg1 ) const`
- `void particles_set_attractors ( RID arg0, int arg1 )`
- `int particles_get_attractors ( RID arg0 ) const`
- `void particles_set_attractor_pos ( RID arg0, int arg1, Vector3 arg2 )`
- `Vector3 particles_get_attractor_pos ( RID arg0, int arg1 ) const`
- `void particles_set_attractor_strength ( RID arg0, int arg1, float arg2 )`
- `float particles_get_attractor_strength ( RID arg0, int arg1 ) const`
- `void particles_set_material ( RID arg0, RID arg1, bool arg2=false )`
- `void particles_set_height_from_velocity ( RID arg0, bool arg1 )`
- `bool particles_has_height_from_velocity ( RID arg0 ) const`
- `RID light_create ( int arg0 )`
- `int light_get_type ( RID arg0 ) const`
- `void light_set_color ( RID arg0, int arg1, Color arg2 )`
- `Color light_get_color ( RID arg0, int arg1 ) const`
- `void light_set_shadow ( RID arg0, bool arg1 )`
- `bool light_has_shadow ( RID arg0 ) const`
- `void light_set_volumetric ( RID arg0, bool arg1 )`

- `bool light_is_volumetric ( RID arg0 ) const`
- `void light_set_projector ( RID arg0, RID arg1 )`
- `RID light_get_projector ( RID arg0 ) const`
- `void light_set_var ( RID arg0, int arg1, float arg2 )`
- `float light_get_var ( RID arg0, int arg1 ) const`
- `RID skeleton_create ( )`
- `void skeleton_resize ( RID arg0, int arg1 )`
- `int skeleton_get_bone_count ( RID arg0 ) const`
- `void skeleton_bone_set_transform ( RID arg0, int arg1, Transform arg2 )`
- `Transform skeleton_bone_get_transform ( RID arg0, int arg1 )`
- `RID room_create ( )`
- `void room_set_bounds ( RID arg0, Dictionary arg1 )`
- `Dictionary room_get_bounds ( RID arg0 ) const`
- `RID portal_create ( )`
- `void portal_set_shape ( RID arg0, Vector2Array arg1 )`
- `Vector2Array portal_get_shape ( RID arg0 ) const`
- `void portal_set_enabled ( RID arg0, bool arg1 )`
- `bool portal_is_enabled ( RID arg0 ) const`
- `void portal_set_disable_distance ( RID arg0, float arg1 )`
- `float portal_get_disable_distance ( RID arg0 ) const`
- `void portal_set_disabled_color ( RID arg0, Color arg1 )`
- `Color portal_get_disabled_color ( RID arg0 ) const`
- `RID camera_create ( )`
- `void camera_set_perspective ( RID arg0, float arg1, float arg2, float arg3 )`
- `void camera_set_orthogonal ( RID arg0, float arg1, float arg2, float arg3 )`
- `void camera_set_transform ( RID arg0, Transform arg1 )`
- `RID viewport_create ( )`
- `void viewport_set_rect ( RID arg0, Rect2 arg1 )`
- `Rect2 viewport_get_rect ( RID arg0 ) const`
- `void viewport_attach_camera ( RID arg0, RID arg1=RID() )`
- `RID viewport_get_attached_camera ( RID arg0 ) const`
- `RID viewport_get_scenario ( RID arg0 ) const`
- `void viewport_attach_canvas ( RID arg0, RID arg1 )`
- `void viewport_remove_canvas ( RID arg0, RID arg1 )`
- `void viewport_set_global_canvas_transform ( RID arg0, Matrix32 arg1 )`
- `RID scenario_create ( )`

- void **scenario\_set\_debug** ( *RID* arg0, *int* arg1 )
- *RID* **instance\_create** ( )
- *RID* **instance\_get\_base** ( *RID* arg0 ) const
- *RID* **instance\_get\_base\_aabb** ( *RID* arg0 ) const
- void **instance\_set\_transform** ( *RID* arg0, *Transform* arg1 )
- *Transform* **instance\_get\_transform** ( *RID* arg0 ) const
- void **instance\_attach\_object\_instance\_ID** ( *RID* arg0, *int* arg1 )
- *int* **instance\_get\_object\_instance\_ID** ( *RID* arg0 ) const
- void **instance\_attach\_skeleton** ( *RID* arg0, *RID* arg1 )
- *RID* **instance\_get\_skeleton** ( *RID* arg0 ) const
- void **instance\_set\_room** ( *RID* arg0, *RID* arg1 )
- *RID* **instance\_get\_room** ( *RID* arg0 ) const
- void **instance\_set\_exterior** ( *RID* arg0, *bool* arg1 )
- *bool* **instance\_is\_exterior** ( *RID* arg0 ) const
- *Array* **instances\_cull\_aabb** ( *AABB* arg0, *RID* arg1 ) const
- *Array* **instances\_cull\_ray** ( *Vector3* arg0, *Vector3* arg1, *RID* arg2 ) const
- *Array* **instances\_cull\_convex** ( *Array* arg0, *RID* arg1 ) const
- *RID* **instance\_geometry\_override\_material\_param** ( *RID* arg0 ) const
- *RID* **instance\_geometry\_get\_material\_param** ( *RID* arg0 ) const
- *RID* **get\_test\_cube** ( )
- *RID* **canvas\_create** ( )
- *RID* **canvas\_item\_create** ( )
- void **canvas\_item\_set\_parent** ( *RID* arg0, *RID* arg1 )
- *RID* **canvas\_item\_get\_parent** ( *RID* arg0 ) const
- void **canvas\_item\_set\_transform** ( *RID* arg0, *Matrix32* arg1 )
- void **canvas\_item\_set\_custom\_rect** ( *RID* arg0, *bool* arg1, *Rect2* arg2 )
- void **canvas\_item\_set\_clip** ( *RID* arg0, *bool* arg1 )
- void **canvas\_item\_set\_opacity** ( *RID* arg0, *float* arg1 )
- *float* **canvas\_item\_get\_opacity** ( *RID* arg0, *float* arg1 ) const
- void **canvas\_item\_set\_self\_opacity** ( *RID* arg0, *float* arg1 )
- *float* **canvas\_item\_get\_self\_opacity** ( *RID* arg0, *float* arg1 ) const
- void **canvas\_item\_set\_z** ( *RID* arg0, *int* arg1 )
- 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\_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\_add\_style\_box** ( *RID* arg0, *Rect2* arg1, *RID* arg2, *RealArray* arg3, *Color* arg4=Color(1,1,1,1) )
- void **canvas\_item\_add\_circle** ( *RID* arg0, *Vector2* arg1, *float* arg2, *Color* arg3 )
- void **viewport\_set\_canvas\_transform** ( *RID* arg0, *RID* arg1, *Matrix32* arg2 )
- void **canvas\_item\_clear** ( *RID* arg0 )
- void **canvas\_item\_raise** ( *RID* arg0 )
- void **cursor\_set\_rotation** ( *float* arg0, *int* arg1 )
- void **cursor\_set\_texture** ( *RID* arg0, *Vector2* arg1, *int* arg2 )
- void **cursor\_set\_visible** ( *bool* arg0, *int* arg1 )
- void **cursor\_set\_pos** ( *Vector2* arg0, *int* arg1 )
- void **black\_bars\_set\_margins** ( *int* left, *int* top, *int* right, *int* bottom )
- void **black\_bars\_set\_images** ( *RID* left, *RID* top, *RID* right, *RID* bottom )
- *RID* **make\_sphere\_mesh** ( *int* arg0, *int* arg1, *float* arg2 )
- void **mesh\_add\_surface\_from\_planes** ( *RID* arg0, *Array* arg1 )
- void **draw** ( )
- void **sync** ( )
- void **free** ( *RID* arg0 )
- void **set\_default\_clear\_color** ( *Color* arg0 )
- *int* **get\_render\_info** ( *int* arg0 )

## 9.351 VScrollBar

**Inherits:** *ScrollBar* < *Range* < *Control* < *CanvasItem* < *Node* < *Object*

**Category:** Core

### 9.351.1 Brief Description

Vertical version of *ScrollBar*, which goes from left (min) to right (max).

## 9.352 VSeparator

**Inherits:** *Separator* < *Control* < *CanvasItem* < *Node* < *Object*

**Category:** Core

### 9.352.1 Brief Description

Vertical version of *Separator*.

## 9.352.2 Description

Vertical version of [Separator](#). It is used to separate objects horizontally, though (but it looks vertical!).

## 9.353 VSlider

**Inherits:** [Slider](#) < [Range](#) < [Control](#) < [CanvasItem](#) < [Node](#) < [Object](#)

**Category:** Core

### 9.353.1 Brief Description

Vertical slider.

### 9.353.2 Description

Vertical slider. See [Slider](#). This one goes from left (min) to right (max).

## 9.354 VSplitContainer

**Inherits:** [SplitContainer](#) < [Container](#) < [Control](#) < [CanvasItem](#) < [Node](#) < [Object](#)

**Category:** Core

### 9.354.1 Brief Description

Vertical split container.

### 9.354.2 Description

Vertical split container. See [SplitContainer](#). This goes from left to right.

## 9.355 WeakRef

**Inherits:** [Reference](#) < [Object](#)

**Category:** Core

### 9.355.1 Brief Description

### 9.355.2 Member Functions

<a href="#">Object</a>	<a href="#">get_ref () const</a>
------------------------	----------------------------------

### 9.355.3 Member Function Description

- *Object* `get_ref()` const

## 9.356 WindowDialog

**Inherits:** *Popup < Control < CanvasItem < Node < Object*

**Inherited By:** *AcceptDialog*

**Category:** Core

### 9.356.1 Brief Description

Base class for window dialogs.

### 9.356.2 Member Functions

<code>void</code>	<code>set_title( String title )</code>
<code>String</code>	<code>get_title() const</code>
<code>TextureButton</code>	<code>get_close_button()</code>

### 9.356.3 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.

### 9.356.4 Member Function Description

- `void set_title( String title )`

Set the title of the window.

- `String get_title() const`

Return the title of the window.

- `TextureButton get_close_button()`

Return the close *TextureButton*.

## 9.357 World

**Inherits:** *Resource < Reference < Object*

**Category:** Core

### 9.357.1 Brief Description

Class that has everything pertaining to a world.

## 9.357.2 Member Functions

<i>RID</i>	<code>get_space () const</code>
<i>RID</i>	<code>get_scenario () const</code>
<i>RID</i>	<code>get_sound_space () const</code>
void	<code>set_environment ( <i>Environment</i> env )</code>
<i>Environment</i>	<code>get_environment () const</code>
<i>PhysicsDirectSpaceState</i>	<code>get_direct_space_state ()</code>

## 9.357.3 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.

## 9.357.4 Member Function Description

- *RID* `get_space () const`
- *RID* `get_scenario () const`
- *RID* `get_sound_space () const`
- void `set_environment ( Environment env )`
- *Environment* `get_environment () const`
- *PhysicsDirectSpaceState* `get_direct_space_state ()`

## 9.358 World2D

**Inherits:** *Resource* < *Reference* < *Object*

**Category:** Core

### 9.358.1 Brief Description

Class that has everything pertaining to a 2D world.

## 9.358.2 Member Functions

<i>RID</i>	<code>get_canvas ()</code>
<i>RID</i>	<code>get_space ()</code>
<i>RID</i>	<code>get_sound_space ()</code>
<i>Physics2DDirectSpaceState</i>	<code>get_direct_space_state ()</code>

## 9.358.3 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.

## 9.358.4 Member Function Description

- *RID* `get_canvas()`
- *RID* `get_space()`
- *RID* `get_sound_space()`
- *Physics2DDirectSpaceState* `get_direct_space_state()`

## 9.359 WorldEnvironment

Inherits: *Spatial* < *Node* < *Object*

Category: Core

### 9.359.1 Brief Description

### 9.359.2 Member Functions

void	<code>set_environment ( Environment env )</code>
<i>Environment</i>	<code>get_environment () const</code>

### 9.359.3 Member Function Description

- void `set_environment ( Environment env )`
- *Environment* `get_environment () const`

## 9.360 XMLParser

Inherits: *Reference* < *Object*

Category: Core

### 9.360.1 Brief Description

### 9.360.2 Member Functions

<i>int</i>	<code>read ()</code>
<i>int</i>	<code>get_node_type ()</code>
<i>String</i>	<code>get_node_name () const</code>
<i>String</i>	<code>get_node_data () const</code>
<i>int</i>	<code>get_node_offset () const</code>
<i>int</i>	<code>get_attribute_count () const</code>
<i>String</i>	<code>get_attribute_name ( int idx ) const</code>
<i>String</i>	<code>get_attribute_value ( int idx ) const</code>
<i>bool</i>	<code>has_attribute ( String name ) const</code>
<i>String</i>	<code>get_named_attribute_value ( String name ) const</code>
<i>String</i>	<code>get_named_attribute_value_safe ( String name ) const</code>
<i>bool</i>	<code>is_empty () const</code>
<i>int</i>	<code>get_current_line () const</code>
<i>void</i>	<code>skip_section ()</code>
<i>int</i>	<code>seek ( int pos )</code>
<i>int</i>	<code>open ( String file )</code>
<i>int</i>	<code>open_buffer ( RawArray buffer )</code>

### 9.360.3 Numeric Constants

- `NODE_NONE = 0`
- `NODE_ELEMENT = 1`
- `NODE_ELEMENT_END = 2`
- `NODE_TEXT = 3`
- `NODE_COMMENT = 4`
- `NODE_CDATA = 5`
- `NODE_UNKNOWN = 6`

### 9.360.4 Member Function Description

- `int read ()`
- `int get_node_type ()`
- `String get_node_name () const`
- `String get_node_data () const`
- `int get_node_offset () const`
- `int get_attribute_count () const`
- `String get_attribute_name ( int idx ) const`
- `String get_attribute_value ( int idx ) const`
- `bool has_attribute ( String name ) const`
- `String get_named_attribute_value ( String name ) const`

- `String get_named_attribute_value_safe ( String name ) const`
- `bool is_empty () const`
- `int get_current_line () const`
- `void skip_section ()`
- `int seek ( int pos )`
- `int open ( String file )`
- `int open_buffer ( RawArray buffer )`

## 9.361 YSort

**Inherits:** `Node2D < CanvasItem < Node < Object`

**Category:** Core

### 9.361.1 Brief Description

### 9.361.2 Member Functions

<code>void</code>	<code>set_sort_enabled ( bool enabled )</code>
<code>bool</code>	<code>is_sort_enabled () const</code>

### 9.361.3 Member Function Description

- `void set_sort_enabled ( bool enabled )`
- `bool is_sort_enabled () const`



---

## Languages

---

## 10.1 GDScript

### 10.1.1 Introduction

GDScript is a high level, dynamically typed programming language used to create content. It uses a syntax that is very similar to the Python language (blocks are indent-based) and its goal is to be very optimal and tightly integrated with the engine, allowing great flexibility for content creation and integration.

### 10.1.2 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 Godot would work more optimally by using a built-in scripting language, as the following barriers were met:

- 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 resulted in duplicate effort and bugs.
- Binding code is always messy and results in several failure points, unexpected bugs and generally low maintainability.

Finally, GDScript was written as a custom solution. The language and interpreter for it ended up being smaller than the binding code itself for Lua and Squirrel, and equally as functional. With time, having a built-in language has proven to be a huge advantage.

### 10.1.3 Example

Some people can learn better by just taking a look at the syntax, so here's a simple example of how it looks.

```
# 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"

# 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

# subclass

class Something:
    var a = 10

# constructor

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, it is advised you read this tutorial: [[GDScript (More Efficiently)]].

## 10.1.4 Language

### 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.

### Operators

The following is the list of supported operators and their precedence (TODO, change since this was made to reflect python operators)

Operator	Description
x[index]	Subscription, Highest Priority
x.attribute	Attribute Reference
extends	Instance Type Checker
~	Bitwise NOT
-x	Negative
* / %	Multiplication / Division / Remainder
+ -	Addition / Subtraction
<< >>	Bit Shifting
&	Bitwise AND
^	Bitwise XOR
&#124;	Bitwise OR
< > == != >= <=	Comparisons
in	Content Test
! not	Boolean NOT
and &&	Boolean AND
or &#124; &#124;	Boolean OR
= += -= *= /= %= &= &#124; =	Assignment, Lowest Priority

### Literals

Literal	Type
45	Base 10 integer
0x8F51	Base 16 (hex) integer
3.14, 58.1e-10	Floating point number (real)
"Hello", "Hi"	Strings
"""Hello, Dude"""	Multiline string
&#64; "Node/Label"	NodePath or StringName

## Comments

Anything from a # to the end of the line is ignored and is considered a comment.

```
# This is a comment
```

Multi-line comments can be created using “”” (three quotes in a row) at the beginning and end of a block of text.

```
""" Everything on these  
lines is considered  
a comment """
```

## 10.1.5 Built-In Types

### Basic Built-In Types

A variable in GDScript can be assigned to several built-in types.

#### null

null is a data type that contains no information, nothing assigned, and it's just empty. It can only be set to one value: null.

#### 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.

### 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: `pos` and `size`. Alternatively contains an `end` field which is `pos+size`. As an alias of this type, `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 `basis` and a `Vector3` field `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 objects. Its size can be changed to anything and starts from index 0.

```
var arr=[]
arr=[1, 2, 3]
arr[0] = "Hi!"
```

Arrays are allocated linearly in memory, so they are fast, but very large arrays (more than tens of thousands of elements) may cause fragmentation.

There are specialized arrays (listed below) for some built-in data types which do not suffer from this and use less memory, but they are atomic and generally run a little slower, so they are only justified for very large amount of data.

### Dictionary

Associative container which contains values referenced by unique keys.

```
var d={4:5, "a key":"a value", 28:[1,2,3]}
d["Hi!"] = 0
```

Lua-style table syntax is also supported, given that it's easier to write and read:

```
var d = {
    somekey = 2,
    otherkey = [2,3,4],
    morekey = "Hello"
}
```

## ByteArray

An array of bytes can only contain bytes (integers from 0 to 255).

This, and all of the following specialized array types, are optimized for memory usage and can't fragment the memory.

## IntArray

Array of integers can only contain integers.

## FloatArray

Array of floats can only contain floats.

## StringArray

Array of strings can only contain strings.

## Vector2Array

Array of Vector2 can only contain 2D Vectors.

## Vector3Array

Array of Vector3 can only contain 3D Vectors.

## ColorArray

Array of Color can only contains colors.

## 10.1.6 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.

```
var a # data type is null by default
var b = 5
var c = 3.8
var 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.

```
const a = 5
const b = Vector2(20, 20)
const c = 10 + 20 # constant expression
const d = Vector2(20, 30).x # constant expression: 20
const e = [1, 2, 3, 4][0] # constant expression: 1
const f = sin(20) # sin() can be used in constant expressions
const g = x + 20 # invalid; this is not a constant expression!
```

## Functions

Functions always belong to a class. The scope priority for variable look-up is: local→class member→global. `self` is provided as an option for accessing class members, but is not always required (and must *not* be defined as the first parameter, like in Python). For performance reasons, functions are not considered class members, so they can't be referenced directly. A function can return at any point. The default return value is null.

```
func myfunction(a, b):
    print(a)
    print(b)
    return a + b # return is optional; without it null is returned
```

## 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 statements is 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.

```
if [expression]:
    statement(s)
elif [expression]:
    statement(s)
else:
    statement(s)
```

### while

Simple loops are created by using *while* syntax. Loops can be broken using *break* or continued using *continue*:

```
while [expression]:
    statement(s)
```

### for

To iterate through a range, such as an array or table, a *for* loop is used. For loops store the index in the loop variable on each iteration.

```

for i in [0, 1, 2]:
    statement # loop iterates 3 times with i as 0, then 1 and finally 2

var dict = {"a":0, "b":1, "c":2}
for i in dict:
    print(dict[i]) # loop iterates the keys; with i being "a", "b" and "c" it prints 0, 1 and 2.

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

```

## Function Call on Base Class

To call a function on a base class (that was overridden in the current one), prepend `.` to the function name:

```
.basefunc()
```

However, remember that functions such as `_init`, and most notifications such as `_enter_tree`, `_exit_tree`, `_process`, `_fixed_process`, etc. are called in all base classes automatically, so this should be only for calling functions you write yourself.

## 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.

## Class File Example

Imagine the following being stored in a file like `myclass.gd`.

```

var a = 5

func print_value_of_a():
    print(a)

```

## Inheritance

A class file can inherit from a global class, another file or a subclass inside another file. Multiple inheritance is not allowed. The `extends` syntax is used. Follows is 3 methods of using `extends`:

```
# extend from some class (global)
extends SomeClass
```

```
# optionally, extend from another file
extends "somefile.gd"
```

```
# extend from a subclass in another file
extends "somefile.gd".Subclass
```

## Inheritance Testing

It's possible to check if an instance inherits from a given class. For this the `extends` keyword can be used as an operator instead:

```
const enemy_class = preload("enemy.gd") # cache the enemy class

# [...]

if (entity extends enemy_class):
    entity.apply_damage()
```

## Constructor

A class can have an optional constructor; a function named `_init` that is called when the class is instanced.

### Arguments to Parent Constructor

When inheriting, parent constructors are called automatically (no need to call `._init()`). If a parent constructor takes arguments, they are passed like this:

```
func _init(args) .(parentargs):
    pass
```

## Sub Classes

A class file can have subclasses. This syntax should be straightforward:

```
class SomeSubClass:
    var a = 5
    func print_value_of_a():
        print(a)

func _init():
    var sc = SomeSubClass.new() #instance by calling built-in new
    sc.print_value_of_a()
```

## Classes as Objects

It may be desired at some point to load a class from a file and then instance it. Since the global scope does not exist, classes must be loaded as a resource. Instancing is done by calling the `new` function in a class object:

```
# load the class (loaded every time the script is instanced)
var MyClass = load("myclass.gd")

# alternatively, using the preload() function preloads the class 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 a scene. If class members have initializers to constant expressions, they will be available for editing in the property editor. Exporting is done by using the `export` keyword:

```
extends Button

export var data # value will be saved
export var number = 5 # also available to the property editor
```

One of the fundamental benefits of exporting member variables is to have them visible in the property 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 for more detail in the exported variables:

```
# if the exported value assigns a constant or constant expression, the type will be inferred and used

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

# integers and strings hint enumerated values

export(int, "Warrior", "Magician", "Thief") var character_class # (editor will set them as 0, 1 and 2)
export(String, "Rebecca", "Mary", "Leah") var character_name

# strings as paths

export(String, FILE) var f # string is a path to a file
export(String, DIR) var f # string is a path to a directory
export(String, FILE, "*.txt") var f # string is a path to a file, custom filter provided as hint

# using paths in the global filesystem is also possible, but only in tool scripts (see further below)

export(String, FILE, GLOBAL, "*.png") var tool_image # string is a path to a PNG file in the global filesystem
export(String, DIR, GLOBAL) var tool_dir # string is a path to a directory in the global filesystem

# multiline strings

export(String, MULTILINE) var text # display a large window to edit strings with multiple lines

# integers and floats hint ranges

export(int, 20) var i # 0 to 20 allowed
export(int, -10, 20) var j # -10 to 20 allowed
export(float, -10, 20, 0.2) var k # -10 to 20 allowed, with stepping of 0.2
export(float, EXP, 100, 1000, 20) var l # exponential range, editing this property using the slider
```

```
# floats with easing hint

export(float, EASE) var transition_speed # display a visual representation of the ease()
function wh

# color can hint availability of alpha

export(Color, RGB) var col  # Color is RGB
export(Color, RGBA) var col  # Color is RGBA
```

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:

```
export(int, FLAGS) var spell_elements = ELEMENT_WIND | ELEMENT_WATER # individually edit the bits of
```

Restricting the flags to a certain number of named flags is also possible. The syntax is very similar to the enumeration syntax:

```
export(int, FLAGS, "Fire", "Water", "Earth", "Wind") var spell_elements = 0 # set any of the given f
```

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 too but there is a restriction. While regular arrays are created local to every 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.

```
# 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]
```

## 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:

```
static func sum2(a, b):
    return a + b
```

## Setters/Getters

It is often useful to know when a member variable changed. It may also be desired to encapsulate its access. For this, GDScript provides a `setter/_getter` helper using the `setget` keyword.

Just add it at the end of the variable definition line like this:

```
var myinteger = 5 setget myinteger_changed
```

If the value of `myinteger` is modified *externally* (not from local usage in the class), the `setter` function will be called beforehand. The `setter` must, then, decide what to do with the new value. The `setter` function looks like this:

```
func myinteger_changed(newvalue):
    myinteger=newvalue
```

A `setter` and a `getter` can be used together too, just define both of them:

```
var myvar setget myvar_set,myvar_get

func myvar_set(newvalue):
    myvar=newvalue

func myvar_get():
    return myvar # getter must return a value
```

Using simply a `getter` is possible too, just skip the setter:

```
var myvar setget ,myvar_get
```

This is especially useful when exporting variables to editor in tool scripts or plugins, for validating input.

Note: As mentioned before, local access will not trigger the setter and getter. For example:

```
func _init():
#does not trigger setter/getter
    myinteger=5
    print(myinteger)
#triggers 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 [[Class: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 [[Class: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.

## Function References

Functions can't be referenced because they are not treated as class members. There are two alternatives to this, though. The `call` function or the `funcref` helper.

```
instance.call("funcname", args) # call a function by name

var fr = funcref(instance, "funcname") # create a function ref
fr.call_func(args)
```

## 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, in the body of the class, just write:

```
# 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_with_args", self, "callback_no_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:

```
func _at_some_func():
    emit_signal("your_signal_name")
    emit_signal("your_signal_name_with_args", 55, 128)
    someinstance.emit_signal("somesignal")
```

## Coroutines

GDScript has some support for coroutines via the `yield` built-in function. The way it works is very simple: 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:

```
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:

```
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:

```
hello  
world  
cheers!
```

## Coroutines & Signals

The real strength of using `yield` is when combined with signals. `yield` can accept two parameters, an object and a signal. When the signal is activated, execution will return. Here are some examples:

```
#resume execution the next frame  
yield( get_tree(), "idle_frame" )  
  
#resume execution when animation is done playing:  
yield( get_node("AnimationPlayer"), "finished" )
```

## 10.2 Using GDScript Efficiently

### 10.2.1 About

This tutorial aims to be a quick reference for how to use GDScript more efficiently. It focuses in common cases specific to the language, but also covers a lot related to using dynamically typed languages.

It's meant to be specially useful for programmers without previous or little experience of dynamically typed languages.

### 10.2.2 Dynamic Nature

#### Pros & Cons of Dynamic Typing

GDScript is a Dynamically Typed language. As such, its main advantages are that:

- 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.
- Run-Time is tiny.
- [[API:Duck-Typing]] and [[API:Polymorphism]] by nature.

While the main cons 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 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 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:

```
int a; // value uninitialized
a=5; // this is valid
a="Hi!"; // this is invalid
```

Dynamic:

```
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:

```
void print_value(int value)
{
    printf("value is %i\n", value);
}

[...]

print_value(55); // valid
print_value("Hello"); // invalid
```

Dynamic:

```
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 later 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++

```
void use_class(SomeClass *instance) {  
  
    instance->use();  
}  
  
void do_something() {  
  
    SomeClass *instance = new SomeClass; //created as pointer  
    use_class(instance); //pass as pointer  
    delete instance; //otherwise it will leak memory  
}
```

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); //pass as reference  
    //garbage collector will get rid of it when not in  
    //use and freeze your game randomly for a second  
}
```

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  
    use_class(instance) # pass as reference  
    #will be unreferenced and deleted
```

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 [[API:Reference]] (the default if nothing is specified) will be freed when not used, but manual memory management is allowed too if inheriting manually from [[API:Object]].

### 10.2.3 Arrays

Arrays in dynamically typed languages can contain many different mixed datatypes inside and are always dynamic (can be resized at any time). Example:

```
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 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
```

GDScript:

```
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:

```
var array = []
array.append(4)
array.append(5)
array.pop_front()
```

or unordered sets:

```
var a = 20
if a in [10,20,30]:
    print("We have a Winner!")
```

## 10.2.4 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 languages such as Lua will go as far as implementing arrays as dictionaries.

Example of Dictionary:

```
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 an 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

var d = {
    name="john",
    age=22
}

print("Name: ", d.name, " Age: ", d.age) # used "." based indexing

# indexing

d.mother="rebecca" #this doesn't work (use syntax below to add a key:value pair)
d["mother"]="rebecca" #this works
d.name="caroline" # if key exists, assignment does work, this is why it's like a quick struct.
```

## 10.2.5 For & While

Iterating in some statically typed languages can be quite complex:

```
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::const_iterator it = strings.begin() ; it != strings.end() ; it++) {
    std::cout << *it << std::endl;
}

```

This is usually greatly simplified in dynamically typed languages:

```

for s in strings:
    print(s)

```

Container datatypes (arrays and dictionaries) are iterable. Dictionaries allow iterating the keys:

```

for key in dict:
    print(key, " -> ", dict[key])

```

Iterating with indices is also possible:

```

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:

```

for(int i=0;i<10;i++) {}

for(int i=5;i<10;i++) {}

for(int i=5;i<10;i+=2) {}

```

Translate to:

```

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:

```

for(int i=10;i>0;i--) {}

```

becomes

```

for i in range(10,0,-1):

```

## 10.2.6 While

While() loops are the same everywhere:

```
var i=0  
  
while(i
```

## 10.2.7 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:

```
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.

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.

```
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:

```
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.

## 10.3 Shading Language

### 10.3.1 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.

## 10.3.2 Language

### Typing

The language is statically typed and supports only a few operations. Arrays, classes, structures, etc are not supported. Several built-in datatypes are provided:

### Data Types

DataType	Description
<i>void</i>	Void
<i>bool</i>	boolean (true or false)
<i>float</i>	floating point
<i>vec2</i>	2-component vector, float subindices (x,y or r,g )
<i>vec3</i>	3-component vector, float subindices (x,y,z or r,g,b )
<i>vec4, color</i>	4-component vector, float subindices (x,y,z,w or r,g,b,a )
<i>mat2</i>	2x2 matrix, vec3 subindices (x,y)
<i>mat3</i>	3x3 matrix, vec3 subindices (x,y,z)
<i>mat4</i>	4x4 matrix, vec4 subindices (x,y,z,w)
<i>texture</i>	texture sampler, can only be used as uniform
<i>cubemap</i>	cubemap sampler, can only be used as uniform

### Syntax

The syntax is similar to C, with statements ending in ; , and comments as // and /\* \*/.

Example:

```
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, it’s 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));
```

### 10.3.3 Built-In Functions

Several Built-in functions are provided for convenience, listed as follows:

. Function	. Description
float	sin( float )   Sine
float	cos( float )   Cosine
float	tan( float )   Tangent
float	asin( float )   arc-Sine
float	acos( float )   arc-Cosine
float	atan( float )   arc-Tangent

```
| vec_type pow( vec_type, float ) | Power |
| vec_type pow( vec_type, vec_type ) | Power (Vec. Exponent) |
| vec_type exp( vec_type ) | Base-e Exponential |
| vec_type log( vec_type ) | Natural Logarithm |
| vec_type sqrt( vec_type ) | Square Root |
| vec_type abs( vec_type ) | Absolute |
| vec_type sign( vec_type ) | Sign |
| vec_type floor( vec_type ) | Floor |
| vec_type trunc( vec_type ) | Trunc |
| vec_type ceil( vec_type ) | Ceiling |
| vec_type fract( vec_type ) | Fractional |
| vec_type mod( vec_type, vec_type ) | Remainder |
| vec_type min( vec_type, vec_type ) | Minimum |
| vec_type max( vec_type, vec_type ) | Maximum |
| vec_type clamp( vec_type value, vec_type min, vec_type max ) | Clamp to Min-Max |
| vec_type mix( vec_type a, vec_type b, float c ) | Linear Interpolate |
| vec_type mix( vec_type a, vec_type b, vec_type c ) | Linear Interpolate (Vector Coef.) |
| vec_type step( vec_type a, vec_type b ) | ` a[i] < b[i] ? 0.0 : 1.0` |
| vec_type smoothstep( vec_type a, vec_type b, vec_type c ) |
| float length( vec_type ) | Vector Length |
| float distance( vec_type, vec_type ) | Distance between vector. |
| float dot( vec_type, vec_type ) | Dot Product |
| vec3 dot( vec3, vec3 ) | Cross Product |
| vec_type normalize( vec_type ) | Normalize to unit length |
| vec3 reflect( vec3, vec3 ) | Reflect |
| color tex( texture, vec2 ) | Read from a texture in noormalized coords |
| color texcube( texture, vec3 ) | Read from a cubemap |
| color texscreen( vec2 ) | Read from screen (generates a copy) |
```

### 10.3.4 Built-In Variables

Depending on the shader type, several built-in variables are available, listed as follows:

#### Material - VertexShader

```
|. Variable |. Description |
| const vec3 SRC_VERTEX | Model-Space Vertex |
| const vec3 SRC_NORMAL | Model-Space Normal |
| const vec3 SRC_TANGENT | Model-Space Tangent |
| const float SRC_BINORMALF | Direction to Compute Binormal |
| vec3 VERTEX | View-Space Vertex |
| vec3 NORMAL | View-Space Normal |
| vec3 TANGENT | View-Space Tangent |
| vec3 BINORMAL | View-Space Binormal |
| vec2 UV | UV |
| vec2 UV2 | UV2 |
```

```
| color COLOR | Vertex Color |
| out vec4 VAR1 | Varying 1 Output |
| out vec4 VAR2 | Varying 2 Output |
| out float SPEC_EXP | Specular Exponent (for Vertex Lighting) |
| out float POINT_SIZE | Point Size (for points) |
| const mat4 WORLD_MATRIX | Object World Matrix |
| const mat4 INV_CAMERA_MATRIX | Inverse Camera Matrix |
| const mat4 PROJECTION_MATRIX | Projection Matrix |
| const mat4 MODELVIEW_MATRIX | (InvCamera * Projection) |
| const float INSTANCE_ID | Instance ID (for multimesh)|
| const float TIME | Time (in seconds) |
```

## Material - FragmentShader

Variable	Description
const vec3 <i>VERTEX</i>	View-Space vertex
const vec4 <i>POSITION</i>	View-Space Position
const vec3 <i>NORMAL</i>	View-Space Normal
const vec3 <i>TANGENT</i>	View-Space Tangent
const vec3 <i>BINORMAL</i>	View-Space Binormal
const vec3 <i>NORMALMAP</i>	Alternative to NORMAL, use for normal texture output.
const vec3 <i>NORMALMAP_DEPTH</i>	Complementary to the above, allows changing depth of normalmap.
const vec2 <i>UV</i>	UV
const vec2 <i>UV2</i>	UV2
const color <i>COLOR</i>	Vertex Color
const vec4 <i>VAR1</i>	Varying 1
const vec4 <i>VAR2</i>	Varying 2
const vec2 <i>SCREEN_UV</i>	Screen Texture Coordinate (for using with texscreen)
const float <i>TIME</i>	Time (in seconds)
const vec2 <i>POINT_COORD</i>	UV for point, when drawing point sprites.
out vec3 <i>DIFFUSE</i>	Diffuse Color
out vec4 <i>DIFFUSE_ALPHA</i>	Diffuse Color with Alpha (using this sends geometry to alpha pipeline)
out vec3 <i>SPECULAR</i>	Specular Color
out vec3 <i>EMISSION</i>	Emission Color
out float <i>SPEC_EXP</i>	Specular Exponent (Fragment Version)
out float <i>GLOW</i>	Glow
out mat4 <i>INV_CAMERA_MATRIX</i>	Inverse camera matrix, can be used to obtain world coords (see example below).

## Material - LightShader

Variable	Description
const vec3 <i>NORMAL</i>	View-Space normal
const vec3 <i>LIGHT_DIR</i>	View-Space Light Direction
const vec3 <i>EYE_VEC</i>	View-Space Eye-Point Vector
const vec3 <i>DIFFUSE</i>	Material Diffuse Color
const vec3 <i>LIGHT_DIFFUSE</i>	Light Diffuse Color
const vec3 <i>SPECULAR</i>	Material Specular Color
const vec3 <i>LIGHT_SPECULAR</i>	Light Specular Color
const float <i>SPECULAR_EXP</i>	Specular Exponent
const vec1 <i>SHADE_PARAM</i>	Generic Shade Parameter
const vec2 <i>POINT_COORD</i>	Current UV for Point Sprite
out vec2 <i>LIGHT</i>	Resulting Light
const float <i>TIME</i>	Time (in seconds)

## CanvasItem (2D) - VertexShader

```
|. Variable |. Description |
| const vec2 SRC_VERTEX | CanvasItem space vertex. |
| vec2 UV | UV |
| out vec2 VERTEX | Output LocalSpace vertex. |
| out vec2 WORLD_VERTEX | Output WorldSpace vertex. (use this or the one above) |
| color COLOR | Vertex Color |
| out vec4 VAR1 | Varying 1 Output |
| out vec4 VAR2 | Varying 2 Output |
| out float POINT_SIZE | Point Size (for points) |
| const mat4 WORLD_MATRIX | Object World Matrix |
| const mat4 EXTRA_MATRIX | Extra (user supplied) matrix via CanvasItem.draw\_set\_transform\(\). Identity by default. |
| const mat4 PROJECTION_MATRIX | Projection Matrix (model coords to screen).|
| const float TIME | Time (in seconds) |
```

## CanvasItem (2D) - FragmentShader

```
|. Variable |. Description |
| const vec4 SRC_COLOR | Vertex color |
| const vec4 POSITION | Screen Position |
| vec2 UV | UV |
| out color COLOR | Output Color |
| out vec3 NORMAL | Optional Normal (used for 2D Lighting) |
| out vec3 NORMALMAP | Optional Normal in standard normalmap format (flipped y and Z from 0 to 1) |
| out float NORMALMAP_DEPTH | Depth option for above normalmap output, default value is 1.0 |
| const texture TEXTURE | Current texture in use for CanvasItem |
| const vec2 TEXTURE_PIXEL_SIZE | Pixel size for current 2D texture |
| in vec4 VAR1 | Varying 1 Output |
| in vec4 VAR2 | Varying 2 Output |
| const vec2 SCREEN_UV | Screen Texture Coordinate (for using with texscreen) |
```

```
| const vec2 POINT_COORD | Current UV for Point Sprite |
| const float TIME | Time (in seconds) |
```

### CanvasItem (2D) - LightShader

```
|. Variable |. Description |
| const vec4 POSITION | Screen Position |
| in vec3 NORMAL | Input Normal |
| in vec2 UV | UV |
| in color COLOR | Input Color |
| const texture TEXTURE | Current texture in use for CanvasItem |
| const vec2 TEXTURE_PIXEL_SIZE | Pixel size for current 2D texture |
| in vec4 VAR1 | Varying 1 Output |
| in vec4 VAR2 | Varying 2 Output |
| const vec2 SCREEN_UV | Screen Texture Coordinate (for using with texscreen) |
| const vec2 POINT_COORD | Current UV for Point Sprite |
| const float TIME | Time (in seconds) |
| vec2 LIGHT_VEC | Vector from light to fragment, can be modified to alter shadow computation. |
| const float LIGHT_HEIGHT | Height of Light |
| const color LIGHT_COLOR | Color of Light |
| out vec4 LIGHT | Light Output (shader is ignored if this is not used) |
```

## 10.3.5 Examples

Material that reads a texture, a color and multiples them, fragment program:

```
uniform color modulate;
uniform texture source;

DIFFUSE = modulate.rgb * tex(source,UV).rgb;
```

Material that glows from red to white:

```
DIFFUSE = vec3(1,0,0) + vec(1,1,1)*mod(TIME,1.0);
```

Standard Blinn Lighting Shader

```
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:

```
//use reverse multiply because INV_CAMERA_MATRIX is world2cam

vec3 world_normal = NORMAL * mat3(INV_CAMERA_MATRIX);
vec3 world_pos = (VERTEX-INV_CAMERA_MATRIX.w.xyz) * mat3(INV_CAMERA_MATRIX);
```

### 10.3.6 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.

## 10.4 Locales

This is the list of supported locales and variants in the engine. It's based on the Unix standard locale strings:

Locale	Language and Variant
ar	Arabic
ar_AE	Arabic (United Arab Emirates)
ar_BH	Arabic (Bahrain)
ar_DZ	Arabic (Algeria)
ar_EG	Arabic (Egypt)
ar_IQ	Arabic (Iraq)
ar_JO	Arabic (Jordan)
ar_KW	Arabic (Kuwait)
ar_LB	Arabic (Lebanon)
ar LY	Arabic (Libya)
ar_MA	Arabic (Morocco)
ar_OM	Arabic (Oman)
ar_QA	Arabic (Qatar)
ar_SA	Arabic (Saudi Arabia)
ar_SD	Arabic (Sudan)
ar_SY	Arabic (Syria)
ar_TN	Arabic (Tunisia)
ar_YE	Arabic (Yemen)
be	Belarusian
be_BY	Belarusian (Belarus)
bg	Bulgarian
bg_BG	Bulgarian (Bulgaria)
ca	Catalan
ca_ES	Catalan (Spain)
cs	Czech
cs_CZ	Czech (Czech Republic)
da	Danish
da_DK	Danish (Denmark)
de	German
de_AT	German (Austria)
de_CH	German (Switzerland)
de_DE	German (Germany)
de_LU	German (Luxembourg)
el	Greek
el_CY	Greek (Cyprus)
Continued on next page	

Table 10.1 – continued from previous page

Locale	Language and Variant
el_GR	Greek (Greece)
en	English
en_AU	English (Australia)
en_CA	English (Canada)
en_GB	English (United Kingdom)
en_IE	English (Ireland)
en_IN	English (India)
en_MT	English (Malta)
en_NZ	English (New Zealand)
en_PH	English (Philippines)
en SG	English (Singapore)
en_US	English (United States)
en_ZA	English (South Africa)
es	Spanish
es_AR	Spanish (Argentina)
es_BO	Spanish (Bolivia)
es_CL	Spanish (Chile)
es_CO	Spanish (Colombia)
es_CR	Spanish (Costa Rica)
es_DO	Spanish (Dominican Republic)
es_EC	Spanish (Ecuador)
es_ES	Spanish (Spain)
es_GT	Spanish (Guatemala)
es_HN	Spanish (Honduras)
es_MX	Spanish (Mexico)
es_NI	Spanish (Nicaragua)
es_PA	Spanish (Panama)
es_PE	Spanish (Peru)
es_PR	Spanish (Puerto Rico)
es_PY	Spanish (Paraguay)
es_SV	Spanish (El Salvador)
es_US	Spanish (United States)
es_UY	Spanish (Uruguay)
es_VE	Spanish (Venezuela)
et	Estonian
et_EE	Estonian (Estonia)
fi	Finnish
fi_FI	Finnish (Finland)
fr	French
fr_BE	French (Belgium)
fr_CA	French (Canada)
fr_CH	French (Switzerland)
fr_FR	French (France)
fr_LU	French (Luxembourg)
ga	Irish
ga_IE	Irish (Ireland)
hi	Hindi (India)
hi_IN	Hindi (India)
hr	Croatian

Continued on next page

Table 10.1 – continued from previous page

Locale	Language and Variant
hr_HR	Croatian (Croatia)
hu	Hungarian
hu_HU	Hungarian (Hungary)
in	Indonesian
in_ID	Indonesian (Indonesia)
is	Icelandic
is_IS	Icelandic (Iceland)
it	Italian
it_CH	Italian (Switzerland)
it_IT	Italian (Italy)
iw	Hebrew
iw_IL	Hebrew (Israel)
ja	Japanese
ja_JP	Japanese (Japan)
ja_JP_JP	Japanese (Japan,JP)
ko	Korean
ko_KR	Korean (South Korea)
lt	Lithuanian
lt_LT	Lithuanian (Lithuania)
lv	Latvian
lv_LV	Latvian (Latvia)
mk	Macedonian
mk_MK	Macedonian (Macedonia)
ms	Malay
ms_MY	Malay (Malaysia)
mt	Maltese
mt_MT	Maltese (Malta)
nl	Dutch
nl_BE	Dutch (Belgium)
nl_NL	Dutch (Netherlands)
no	Norwegian
no_NO	Norwegian (Norway)
no_NO_NY	Norwegian (Norway,Nynorsk)
pl	Polish
pl_PL	Polish (Poland)
pt	Portuguese
pt_BR	Portuguese (Brazil)
pt_PT	Portuguese (Portugal)
ro	Romanian
ro_RO	Romanian (Romania)
ru	Russian
ru_RU	Russian (Russia)
sk	Slovak
sk_SK	Slovak (Slovakia)
sl	Slovenian
sl_SI	Slovenian (Slovenia)
sq	Albanian
sq_AL	Albanian (Albania)
sr	Serbian

Continued on next page

Table 10.1 – continued from previous page

Locale	Language and Variant
sr_BA	Serbian (Bosnia and Herzegovina)
sr_CS	Serbian (Serbia and Montenegro)
sr_ME	Serbian (Montenegro)
sr_RS	Serbian (Serbia)
sv	Swedish
sv_SE	Swedish (Sweden)
th	Thai
th_TH	Thai (Thailand)
th_TH_TH	Thai (Thailand,TH)
tr	Turkish
tr_TR	Turkish (Turkey)
uk	Ukrainian
uk_UA	Ukrainian (Ukraine)
vi	Vietnamese
vi_VN	Vietnamese (Vietnam)
zh	Chinese
zh_CN	Chinese (China)
zh_HK	Chinese (Hong Kong)
zh_SG	Chinese (Singapore)
zh_TW	Chinese (Taiwan)

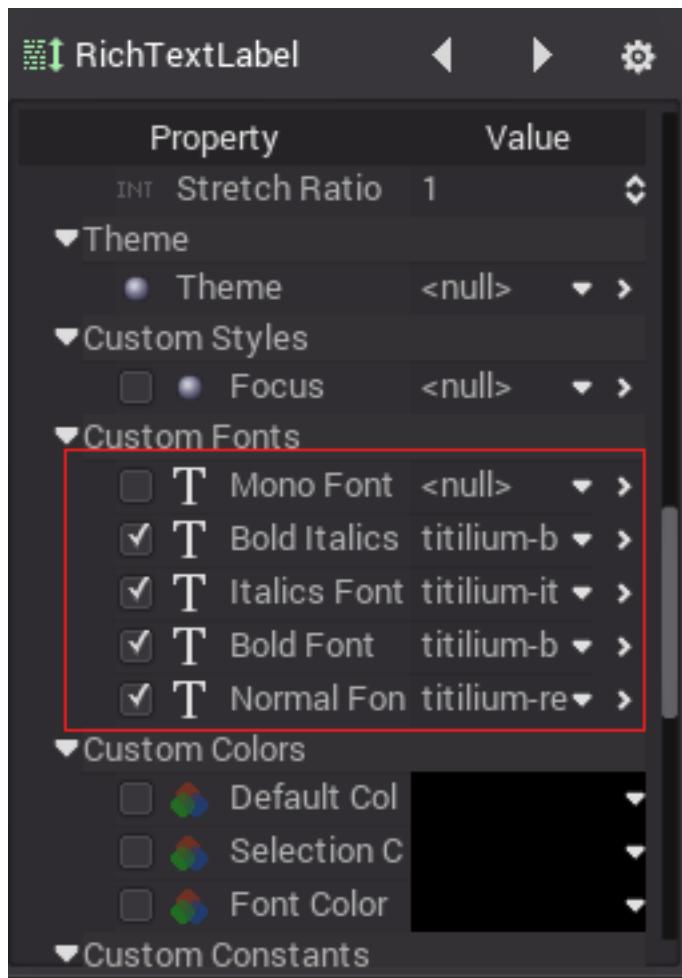
## 10.5 BBCODE RichTextLabel

### 10.5.1 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.

### 10.5.2 Setting Up

For RichTextLabel to work properly, it must be set-up. This means loading the intended fonts in the relevant properties:



### 10.5.3 Reference

Command	Tag	Description
<b>bold</b>	[b]{text}[/b]	Makes {text} bold.
<b>italics</b>	[i]{text}[/i]	Makes {text} italics.
<b>under-line</b>	[u]{text}[/u]	Makes {text} underline.
<b>code</b>	[code]{text}[/code]	Makes {text} monospace.
<b>center</b>	[center]{text}[/center]	Makes {text} centered.
<b>right</b>	[right]{text}[/right]	Makes {text} right-aligned.
<b>fill</b>	[fill]{text}[/fill]	Makes {text} fill width.
<b>indent</b>	[indent]{text}[/indent]	Increase indent level of {text}.
<b>url</b>	[url]{url}[/url]	Show as such.
<b>url (ref)</b>	[url=<url>]{text}[/url]	Makes {text} reference .
<b>image</b>	[img=<path>]{text}[/img]	Insert image at resource .
<b>font</b>	[font=<path>]{text}[/font]	Use custom font at for {text}.
<b>color</b>	[color=&lt;code/name>]{text}[/color]	Change {text} color, use # format such as #ff00ff or name.

## Built-In Color Names

List of valid color names for the [color=] tag:

- aqua
- black
- blue
- fuchsia
- gray
- green
- lime
- maroon
- navy
- purple
- red
- silver
- teal
- white
- yellow



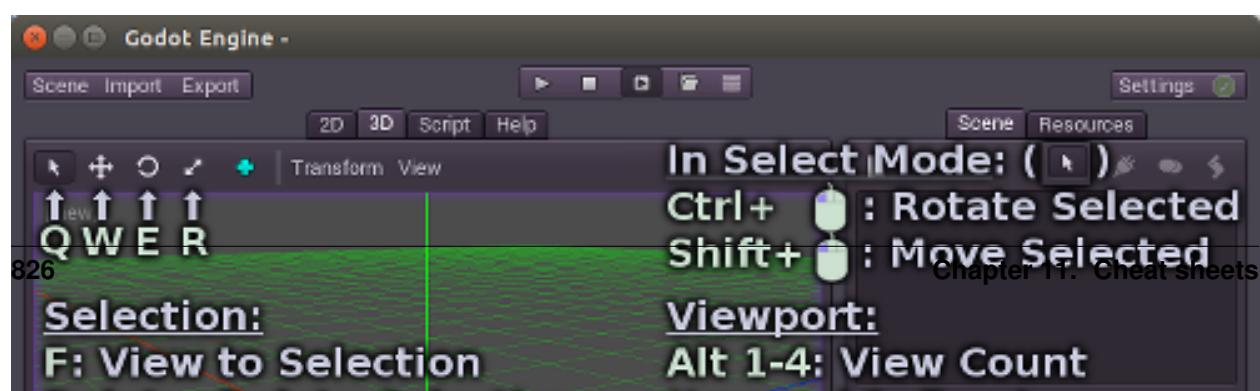
## Cheat sheets

### 11.1 2D and 3D Keybindings

#### 11.1.1 2D Viewport

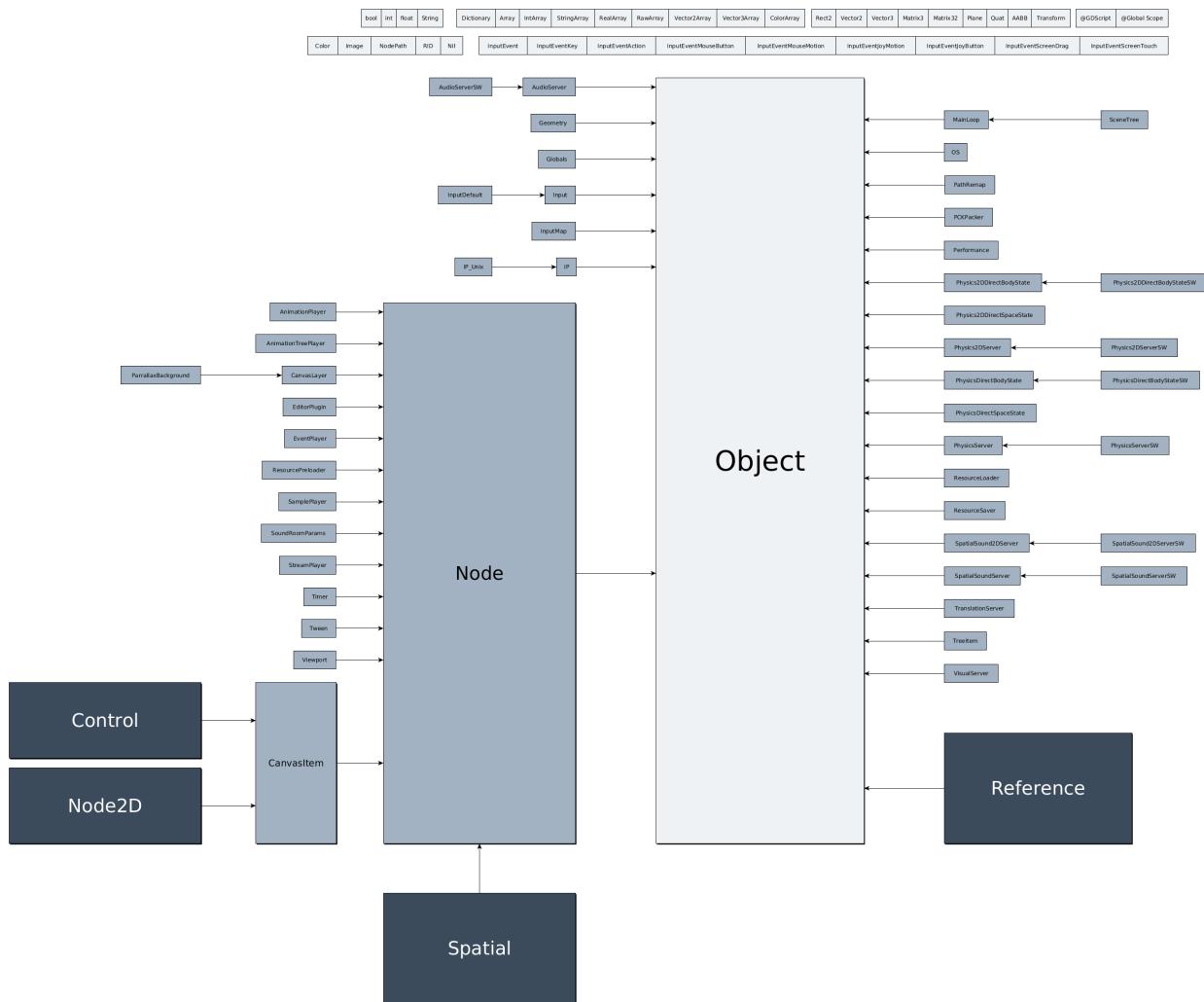


#### 11.1.2 3D Viewport

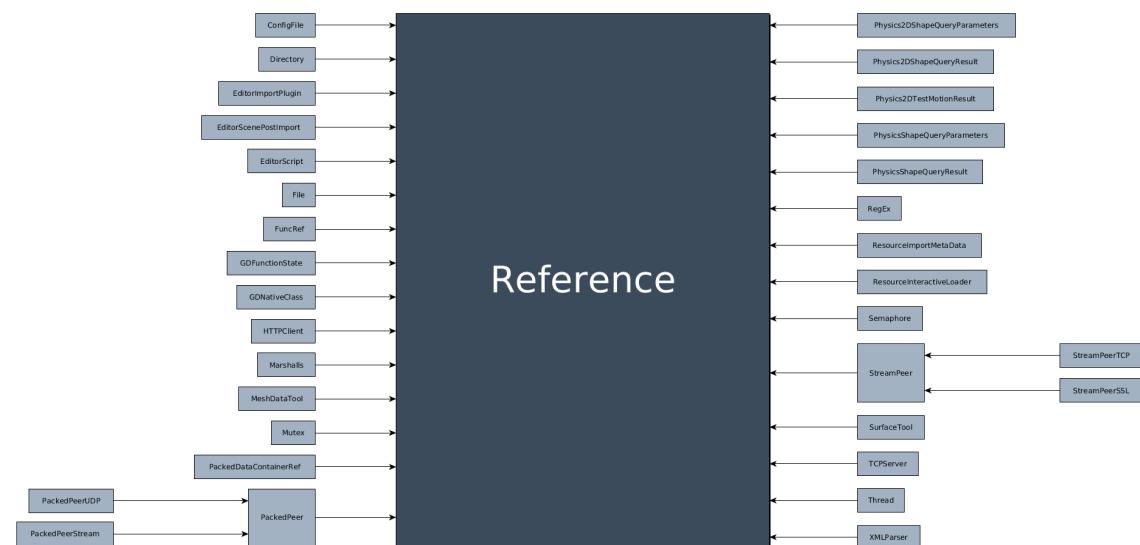




## 11.2.1 Object



## 11.2.2 Reference



attachment:sources.zip



---

## Compiling

---

## 12.1 Introduction to the buildsystem

### 12.1.1 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.

### 12.1.2 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:

```
user@host:~/godot$ scons
scons: Reading SConscript files ...
No valid target platform selected.
The following were detected:
```

```
android
server
javascript
windows
x11

Please run scons again with argument: platform=<string>
scons: done reading SConscript files.
scons: Building targets ...
scons: `.' is up to date.
scons: done building targets.
```

To build 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 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 processes:

```
user@host:~/godot$ scons platform=x11 -j 4
```

### 12.1.3 Resulting binary

The resulting binaries will be placed in the bin/ subdirectory, generally with this naming convention:

```
godot..[opt].[tools/debug]..
```

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 <http://www.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.

### 12.1.4 Tools

Tools are enabled by default in all 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.

```
scons platform= tools=yes/no
```

## 12.1.5 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.

```
scons platform= target=debug/release_debug/release
```

This flag appends ".debug" suffix (for debug), or ".tools" (for debug with tools enables). When optimization is enabled (release) it appends the ".opt" suffix.

## 12.1.6 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.

```
scons platform= bits=default/32/64
```

This flag appends ".32" or ".64" suffixes to resulting binaries when relevant.

## 12.1.7 Export templates

Official export templates are downloaded from the Godot Engine site: <http://www.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
osx.zip
version.txt
windows_32_debug.exe
windows_32_release.exe
windows_64_debug.exe
```

```
windows_64_release.exe  
windows_debug.exe  
windows_release.exe
```

To create those yourself, just follow the instructions detailed for each platform in this same tutorial section. Each platform explains how to create it's 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.

## 12.2 Compiling for Windows

### 12.2.1 Requirements

For compiling under Windows, the following is required:

- [Visual C++](#), Visual C++ Express compiler or Visual Studio Community (recommended) at least the 2010 version (10.0) up to 2015 (14.0). **Make sure you get a version that can compile for C++, Desktop.**
- [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](#).

### 12.2.2 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 and provides a .bat 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.

### 12.2.3 Compiling

Start a Visual Studio command prompt (it sets up environment variables needed by SCons to locate the compiler and SDK), go to the root dir of the engine source code and type:

```
C:\\\\godot> scons platform=windows
```

If all goes well, the resulting binary executable will be placed in C:\\godot\\bin\\godot.windows.tools.exe. This executable file contains the whole engine and runs without any dependencies. Executing it will bring up the project manager.

### 12.2.4 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. The visual studio command prompt calls a .bat file that sets up environment variables (vcvarsall.bat). To build the whole engine from a single command outside the command prompt, the following should be called in a .bat file:

```
C:\\\\path_to_sdk\\\\vcvarsall.bat && scons bin/godot.windows.tools.exe
```

**NOTE:** It seems the latest Visual Studio does not include a desktop command prompt (No, Native tools for x86 is not it). The only way to build it seems to be by running:

```
"C:\Program Files (x86)\Microsoft Visual Studio 12.0\VC\vcvarsall.bat" && c:\python27\scons p=
```

(or however your VS and Scons are installed)

### 12.2.5 Cross compiling

If you are a Linux or Mac user, you need to install mingw32 and mingw-w64. Under Ubuntu or Debian, just run the following commands:

```
apt-get install mingw32 mingw-w64
```

If you are using other distro, scons will check for the following binaries:

```
i586-mingw32msvc-gcc  
i686-w64-mingw32-gcc
```

If the binaries are named or located somewhere else, export the following env variables:

```
export MINGW32_PREFIX="/path/to/i586-mingw32msvc-"  
export MINGW64_PREFIX="/path/to/i686-w64-mingw32-"
```

To make sure you are doing things correctly, executing the following in the shell should result in a working compiler:

```
user@host:~$ ${MINGW32_PREFIX}gcc  
gcc: fatal error: no input files
```

### 12.2.6 Creating Windows export templates

Windows export templates are created by compiling Godot as release, with the following flags:

(for 32 bits, using Mingw32 command prompt or Visual Studio command prompt)

```
C:\\godot> scons platform=windows tools=no target=release bits=32  
C:\\godot> scons platform=windows tools=no target=release_debug bits=32
```

(for 64 bits, using Mingw-w64 or Visual Studio command prompt)

```
C:\\godot> scons platform=windows tools=no target=release bits=64  
C:\\godot> scons platform=windows tools=no target=release_debug bits=64
```

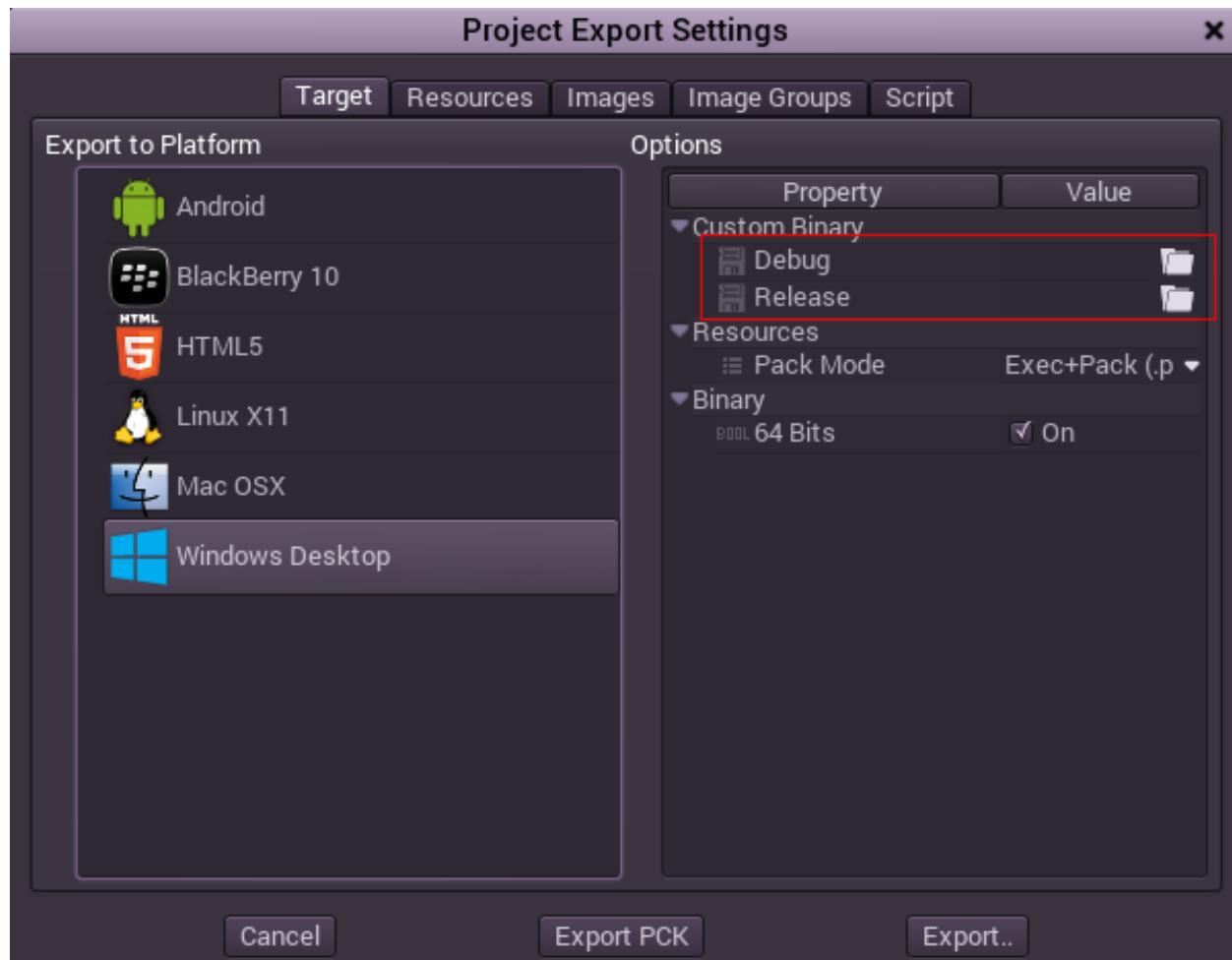
If you plan on replacing the standard templates, copy these to:

```
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:



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.

## 12.3 Compiling for Linux

### 12.3.1 Requirements

For compiling under Linux or other Unix variants, the following is required:

- GCC or LLVM
- Python 2.7+ (3.0 is untested as of now).
- SCons build system.
- X11 and MESA development Libraries
- Xinerama Libraries
- ALSA development libraries
- Freetype (for the editor)
- OpenSSL (for HTTPS and TLS)

- pkg-config (used to detect the above three)
- libevdev-dev and libudev-dev (for facultative joypad support)

For Ubuntu users:

```
apt-get install scons pkg-config libx11-dev libxcursor-dev build-essential libasound2-dev libfreetype
```

If you wish to have Joypad support, libevdev-dev and libudev-dev are required.

```
apt-get install libevdev-dev libudev-dev
```

### 12.3.2 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.

### 12.3.3 Building export templates

To build Linux export templates, run the build system with the following parameters:

(32 bits)

```
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)

```
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 in linux is quite difficult and might need a chroot environment.

In Ubuntu, compilation works without a chroot but some libraries (.so) might be missing from /usr/lib32. Symlinking the missing .so files from /usr/lib results in a working build.

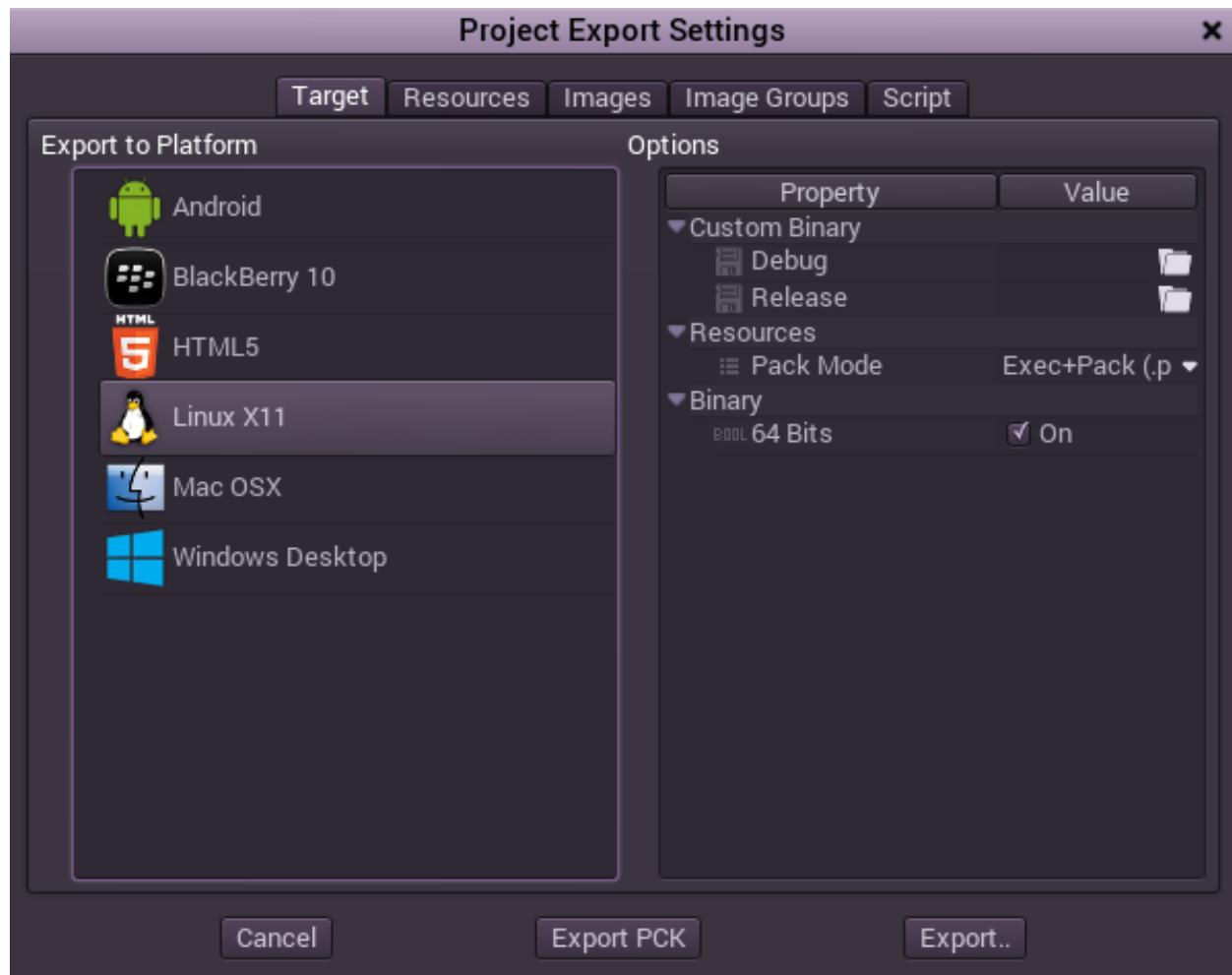
To create standard export templates, the resulting files must be copied to:

```
/home/youruser/.godot/templates
```

and named like this:

```
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.

## 12.4 Compiling for OSX

### 12.4.1 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

### 12.4.2 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. There is a .app template to put the binary into in tools/Godot.app.

### 12.4.3 Cross-compiling

It is possible to compile for OS X in a Linux environment (and maybe also in Windows with Cygwin). For that you will need [OSXCross](#) for being able to use OS X as target. First, follow the instructions to install it:

```
# Clone the OSXCross repository (https://github.com/tpoechtrager/osxcross) somewhere in your machine (or download a Zip file and extract it somewhere). E.g.
```

```
~$ 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.

```
~$ export OSXCROSS_ROOT=/home/myuser/sources/oscross
```

Now you can compile with SCons like you normally would:

```
~/godot$ scons platform=osx
```

## 12.5 Compiling for Android

### 12.5.1 Note

For 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 do all the steps mentioned in the [[Exporting for Android]] tutorial before attempting your custom export template.

### 12.5.2 Requirements

For compiling under Windows, the following is required:

- Python 2.7+ (3.0 is untested as of now).
- SCons build system.
- Android SDK version 8 and 13
- Android NDK

### 12.5.3 Setting Up SCons

Set the environment variable `ANDROID_HOME` to point to the Android SDK.

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, use `export ANDROID_HOME=/path/to/android-sdk` and `export ANDROID_NDK_ROOT=/path/to/android-ndk`.

## 12.5.4 Compiling

Go to the root dir of the engine source code and type:

```
C:\\\\godot> scons platform=android
```

This should result in a regular .so in \bin folder as if it was compiled with flags: `tools=no target=debug`. The resulting file will be huge because it will contain all debug symbols, so for next builds, using `target=release_debug` or `target=release` is recommended.

Copy the .so to the `libs/armeabi` Android folder (or symlink if you are in Linux or OSX). Note: Git does not support empty directories so you will have to create it if it does not exist:

```
C:\\\\godot> mkdir platform/android/java/libs  
C:\\\\godot> mkdir platform/android/java/libs/armeabi
```

Then copy or symlink:

```
C:\\\\godot> copy bin/libgodot.android..so platform/android/java/libs/armeabi/libgodot_android.so
```

alternatively if you are under unix you can symlink:

```
user@host:~/godot$ ln -s bin/libgodot.android..so platform/android/java/libs/armeabi/libgodot_android.so
```

Remember that only *one* of `libgodot_android.so` must exist for each platform, for each build type (release, debug, etc), it must be replaced.

**Note:** The file inside `libs/armeabi` must be renamed to “**libgodot\_android.so**”, or else unsatisfied link error will happen at runtime.

If you also want to include support for x86 Android, add the following compile flag: `x86=yes`, then copy/symlink the resulting folder to the x86 folder:

```
C:\\\\godot> cp bin/libgodot.android..x86.so platform/android/java/libs/x86/libgodot_android.so
```

This will create a fat binary that works in both platforms, but will add about 6 megabytes to the APK.

## 12.5.5 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 numbers, then set the following environment variables:

```
NDK_TOOLCHAIN (by default set to "arm-eabi-4.4.0")  
NDK_TARGET (by default set to "arm-linux-androideabi-4.8")
```

## 12.5.6 Building the APK

To compile the APK, go to the Java folder and run ant, either for debug or release build:

```
C:\\\\godot\\\\platform\\\\android\\\\java> ant debug
```

```
C:\\\\godot\\\\platform\\\\android\\\\java> ant release
```

In the java/bin subfolder, the resulting apk can be used as export template.

**Note:** If you reaaaally feel oldschoold, you can copy your entire game (or symlink) to the assets/ folder of the Java project (make sure engine.cfg is in assets/) and it will work, but you lose all the benefits of the export system (scripts are not byte-compiled, textures not converted to Android compression, etc. so it's not a good idea).

## 12.5.7 Compiling export templates

Godot needs the freshly compiled APK as export templates. It 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).

Compiling the standard export templates is done by calling scons with the following arguments:

(debug)

```
C:\\\\godot> scons platform=android target=release_debug
C:\\\\godot> cp bin/libgodot_android.opt.debug.so platform/android/java/libs/armeabi
C:\\\\godot> cd platform/android/java
C:\\\\godot\\\\platform\\\\android\\\\java> ant release
```

Resulting APK is in:

```
platform/android/java/bin/Godot-release-unsigned.apk
```

(release)

```
C:\\\\godot> scons platform=android target=release
C:\\\\godot> cp bin/libgodot_android.opt.so platform/android/java/libs/armeabi
C:\\\\godot> cd platform/android/java
C:\\\\godot\\\\platform\\\\android\\\\java> ant release
```

Resulting APK is in:

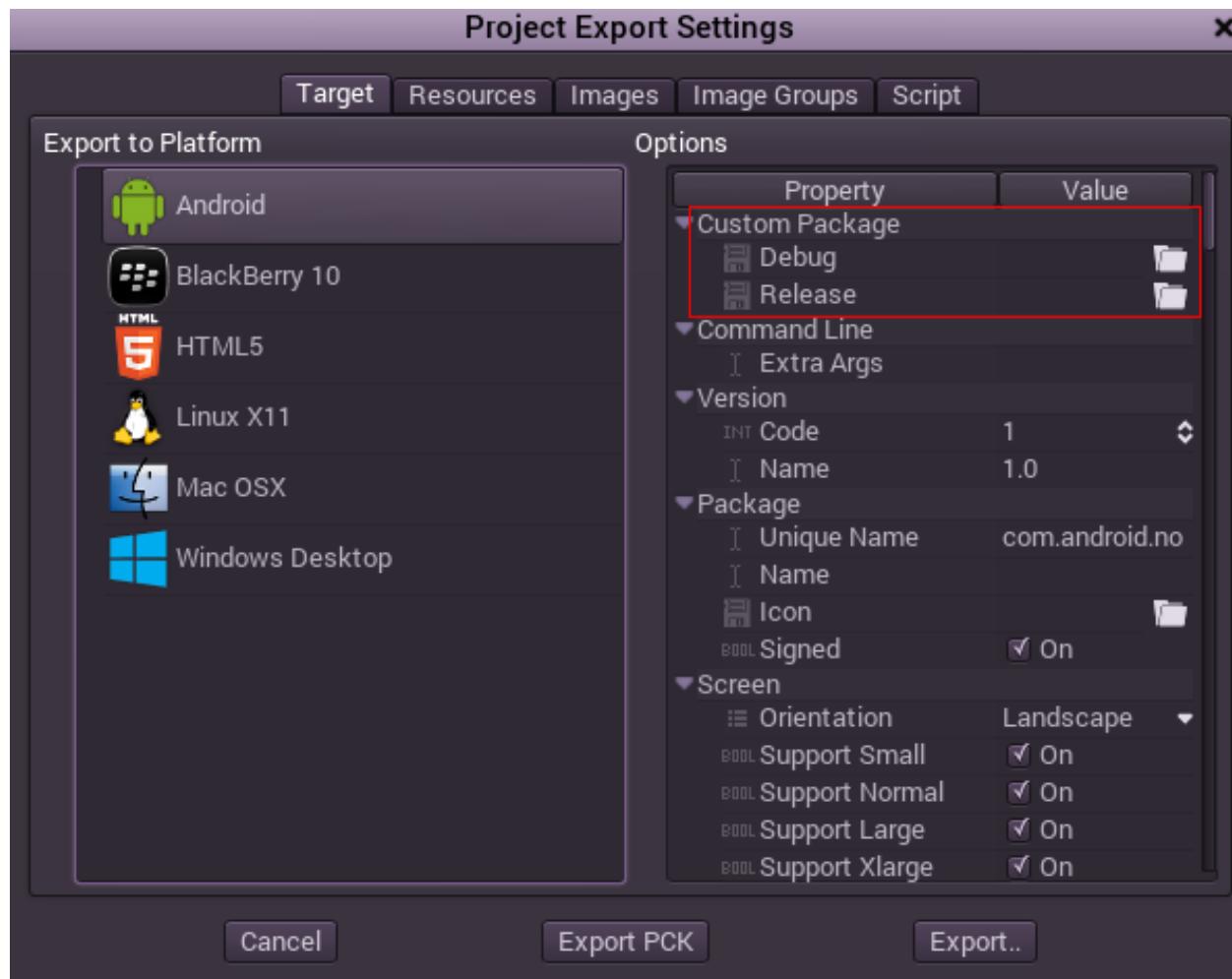
```
platform/android/java/bin/Godot-release-unsigned.apk
```

(same as before)

They must be copied to your templates folder with the following names:

```
android_debug.apk
android_release.apk
```

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:



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 the next time you build you automatically have the custom templates referenced.

## 12.5.8 Troubleshooting

### Application not installed

Android might complain the application is not correctly installed. If so, check the following:

- Check that the debug keystore is properly generated.
- Check that jarsigner is from JDK6.

If it still fails, open a command line and run logcat:

```
C:\\\\android-sdk\\\\platform-tools> adb logcat
```

And check the output while the application is installed. Reason for failure should be presented there. Seek assistance if you can't figure it out.

## Application exits immediately

If the application runs but exits immediately, there might be one of the following reasons:

- libgodot\_android.so is not in libs/armeabi
- Device does not support armv7 (try compiling yourself for armv6)
- Device is Intel, and apk is compiled for ARM.

In any case, `adb logcat` should also show the cause of the error.

## 12.6 Compiling for iOS

### 12.6.1 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.

### 12.6.2 Compiling

Open a Terminal, go to the root dir of the engine source code and type:

```
$ scons p=iphone bin/godot.iphone.debug
```

for a debug build, or:

```
$ scons p=iphone bin/godot.iphone.opt 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=isim bin/godot.isim.tools
```

for a Simulator executable.

### 12.6.3 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.

## 12.7 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.

### 12.7.1 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 do not download any code, which means it should be OK to use the procedure described here and cross-compiling the binary.

### 12.7.2 Requirements

- **XCode with the iOS SDK** (a dmg image)
- **Clang >=3.5** for your development machine installed and in the PATH. It needs to be version  $\geq 3.5$  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**.

### 12.7.3 Configuring the environment

#### **darling-dmg**

# Clone the repository in your machine:

```
$ git clone https://github.com/LubosD/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 have, 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/*
$ cp -r xcode/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/include/c++/* iPhoneOS
$ 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`.

### 12.7.4 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 bits=32 target=release_debug IPHONESDK="/path/to/iPhoneSDK"
$ scons -j 4 platform=iphone bits=64 target=release_debug IPHONESDK="/path/to/iPhoneSDK"
```

IPHONEPATH=  
IPHONEPATH=

## 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.32 bin/godot.ipHONE.opt.debug.64
```

Then you will have an iOS fat binary in `bin/godot.ipHONE.opt.debug.fat`.

## 12.8 Compiling for Universal Windows Apps

This page documents the current state of the “winrt” platform, used to support “Windows Store Apps” for Windows 8.1, and Windows Phone 8.1 apps using Microsoft’s new “Universal” APIs.

### 12.8.1 Requirements

- Windows 8
- SCons (see [[Compiling for Windows]] for more details)
- Visual Studio 2013 for Windows (but *not* “for Windows Desktop”). Tested on “Microsoft Visual Studio Express 2013 for Windows Version 12.0.31101.00 Update 4”.

### 12.8.2 Compiling

The platform can compile binaries for both Windows 8.1 and Windows Phone 8.1. The architecture is decided by the environment variable “PLATFORM”.

#### Windows 8.1

- Open a “VS 2013 x64 Cross Tools Command Prompt”
- The value of environment variable “PLATFORM” should be “x64”
- Run scons with platform=winrt from the root of the source tree:

```
C:\godot_source> scons platform=winrt
```

- You should get an executable file inside bin/ named according to your build options, for the architecture “x64”, for example “godot.winrt.tools.x64.exe”.

#### Windows Phone 8.1

- Open a “Visual Studio 2012 ARM Phone Tools Command Prompt”
- The value of environment variable “PLATFORM” should be “arm”
- Run scons with platform=winrt from the root of the source tree:

```
C:\godot_source> scons platform=winrt
```

- You should get an executable file inside bin/ named according to your build options, for the architecture “arm”, for example “godot.winrt.tools.arm.exe”.

### 12.8.3 Running

On Visual studio, create a new project using any of the “Universal App” templates found under Visual C++ -> Store Apps -> Universal Apps. “Blank App” should be fine.

On the “Solution Explorer” box, you should have 3 sections, “App.Windows (Windows 8.1)”, “App.WindowsPhone (Windows Phone 8.1)” and “App.Shared”. You need to add files to each section:

## App.Shared

Add a folder named “game” containing your game content (can be individual files or your data.pck). Remember to set the “Content” property of each file to “True”, otherwise your files won’t get included in the package.

## App.Windows

- Add your windows executable, and all the .dll files found on platform/winrt/x64/bin on the godot source. Remember to also set the “Content” property.
- Find the file “Package.appxmanifest”. Right click on it and select “Open with...” then “XML (Text) Editor” from the list.
- Find the “Application” section, and add (or modify) the “Executable” property with the name of your .exe. Example:

```
<Application Id="App" Executable="godot.winrt.tools.x64.exe" EntryPoint="App_Windows.App">
```

## App.WindowsPhone

Repeat all the steps from App.Windows, using your arm executable and the dlls found in platform/winrt/arm/bin. Remember to set the “Content” property for all the files.

Use the green “Play” button on the top to run. The drop down menu next to it should let you choose the project (App.Windows or App.WindowsPhone) and the device (“Local Machine”, “Device” for an attached phone, etc).

### 12.8.4 Angle

ANGLE precompiled binaries are provided on platform/winrt/x64 and platform/winrt/arm. They are built from MSOpenTech’s “future-dev” branch, found here: <https://github.com/MSOpenTech/angle>. The visual studio ‘solutions’ used are found on “projects/winrt/windows/angle.sln” and “projects/winrt/windowsphone/angle.sln”.

### 12.8.5 What’s missing

- Audio
- Semaphores
- Keyboard input
- Proper handling of screen rotation
- Proper handling of other events such as focus lost, back button, etc.
- Packaging and deploying to devices from the editor.
- Adding Angle to our tree and compiling it from there. The same source could also be used to build for Windows (and use Angle instead of native GL, which will be more compatible with graphics hardware)

### 12.8.6 Packages

This is what we know:

- App packages are documented here: <http://msdn.microsoft.com/en-us/library/windows/apps/xaml/hh464929.aspx>
- There are 2 command line tools that might be useful, [App Packager](#) and [SignTool](#).

- There are a bunch of tools on “powershell” that deal with packages that might be relevant: <http://technet.microsoft.com/library/dn448373.aspx>
- When running a Windows 8.1 app on “Local Machine” from Visual studio, the app seems to run from an uncompressed directory on the filesystem in an arbitrary location (ie. outside of the proper directory where Apps are installed), but there is some special registry entry made for it, so we know it’s possible to skip the packaging step to run locally (in the case of very big games this can be useful).

## 12.9 Batch building templates

The following is almost the same exact script we use to build all the export templates that go to the site. If you want to build or roll them yourself, this might be of use.

(note: mac stuff is missing)

```
#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
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
ant -s platform/android/java/build.xml release
cp platform/android/java/bin/Godot-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
ant -s platform/android/java/build.xml release
cp platform/android/java/bin/Godot-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
cp bin/godot_server.server.opt.tools.64 release/linux_server.64
upx release/linux_server.64

scons -j 4 p=x11 target=release_debug tools=yes bits=64
cp bin/godot.x11.opt.tools.64 release/godot_x11.64
# upx release/godot_x11.64 -- fails on some linux distros

scons -j 4 p=x11 target=release_debug tools=yes bits=32
cp bin/godot.x11.opt.tools.32 release/godot_x11.32

scons -j 4 p=windows target=release_debug tools=yes bits=64
cp bin/godot.windows.opt.tools.64.exe release/godot_win64.exe
x86_64-w64-mingw32-strip release/godot_win64.exe
#upx release/godot_win64.exe

scons -j 4 p=windows target=release_debug tools=yes bits=32
cp bin/godot.windows.opt.tools.32.exe release/godot_win32.exe
x86_64-w64-mingw32-strip release/godot_win32.exe
#upx release/godot_win64.exe

[...] # mac stuff

# Update classes.xml (used to generate doc)

cp doc/base/classes.xml .
release/linux_server.64 -doctool classes.xml


cd demos
rm -f godot_demos.zip
zip -r godot_demos *
cd .. 

cd tools/export/blender25
zip -r bettercollada *
```

```
mv bettercollada.zip ../../..
cd ..../..
```

## 12.10 Configure an IDE

### 12.10.1 With Eclipse

### 12.10.2 With QtCreator

### 12.10.3 With another editor (vim, emacs, Atom...)



---

## Advanced

---

### 13.1 Developing in C++

#### 13.1.1 Introduction to Godot development

This page introduces the global organization of Godot Engine's source code.

##### Debugging the editor with gdb

If you are writing or correcting bugs affecting Godot Engine editor, remember that the binary runs the launcher first, which runs the editor in another process. Thus, you need to run the editor directly by passing the `-e` argument to Godot Engine editor's binary.

#### 13.1.2 Core types

Godot has a rich set of classes and template that make for its core, and everything is built upon them.

This reference will try to list them in order for their better understanding.

##### 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](#)

## 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.

## 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:

```
memalloc()  
memrealloc()  
memfree()
```

These are equivalent to the usual malloc, realloc, free of the standard library.

For C++-style allocation, special macros are provided:

```
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
```

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:

```
DVector::Read r = dvector.read()
int someint = r[4]
```

and

```
DVector::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

## Containers

Godot provides also a set of common containers:

- Vector
- List
- Set
- Map

The 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:

```
for(List::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

## 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/exracting, as well as helpers for conversion and visualization.

**References:**

- [core/ustring.h](#)

## 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](#)

## Math types

There are several linear math types available in the core/math directory, they are basically just that.

**References:**

- [core/math](#)

## NodePath

This is a special datatype used for storing paths in a scenetree and referencing them fast.

**References:**

- [core/path\\_db.h](#)

## 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](#)

### 13.1.3 Variant class

#### 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 it's atomic/native datatype).
- Be hashed, so it can be compared quickly to other 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](#)

### 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 Nutex should be created to lock if multi-thread access is desired.

#### References:

- [core/dictionary.h](#)
- [core/array.h](#)

### 13.1.4 Object class

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.

```
class CustomObject : public Object {  
    OBJ_TYPE(CustomObject, Object); // this required to inherit  
};
```

This makes objects gain a lot of functionality, like for example

```
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](#)

### Registering an Object

ObjectTypeDB is a static class that hold 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:

```
ObjectTypeDB::register_type()
```

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.

```
ObjectTypeDB::register_virtual_type()
```

Object derived classes can override a 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:

```
ObjectTypeDB::register_method(_MD("methodname", "arg1name", "arg2name"), &MyCustethod);
```

Default values for arguments can be passed in reverse order:

```
ObjectTypeDB::register_method(_MD("methodname", "arg1name", "arg2name"), &MyCustomType::method, DEFVAL(-1))
```

`_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 ignore and a string passing the name can be passed for brevity.

## References:

- core/object\_type\_db.h

## Constants

Classes often have enums such as:

```
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:

```
VARIANT_ENUM_CAST( MyClass::SomeMode ); // now functions that take SomeMode can be bound
```

The constants can also be bound inside `_bind_methods`, by using:

```
BIND_CONSTANT( MODE_FIRST );
BIND_CONSTANT( MODE_SECOND );
```

## 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:

```
PropertyInfo(type, name, hint, hint_string, usage_flags)
```

For example:

```
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 wont be serialized.

or

```
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 hint flags were specified, the default ones are PROPERTY\_USAGE\_STORAGE and PROPERTY\_USAGE\_EDITOR.

There are plenty of hints and usages 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 a 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.

### 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 *r_props); //return list of properties
bool _get(const StringName& p_property, Variant& r_value) const; //return true if property was found
bool _set(const StringName& p_property, const Variant& 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.

### 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).

### Signals

Objects can have a set of signals defined (similar to Delegates in other languages). Connecting to them is rather easy:

```
obj->connect(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:

```
ADD_SIGNAL( MethodInfo("been_killed") )
```

### 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.

```
class MyReference: public Reference {
    OBJ_TYPE( MyReference ,Reference );
};

Ref myref = memnew( MyReference );
```

`myref` is reference counted. It will be freed when no more `Ref<>` templates point to it.

**References:**

- core/reference.h

**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

**Resource loading**

Resources can be loaded with the ResourceLoader API, like this:

```
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

**Resource saving**

Saving a resource can be done with the resource saver API:

```
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

### 13.1.5 Custom modules in C++

#### 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 them modules/ subdirectory of the build system. By default, two modules exist, GDScript (which, yes it's 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.

#### 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++.

#### 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 wiki 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):

```
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:

```
/* 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.

```
/* sumator.cpp */

#include "sumator.h"

void Sumator::add(int value) {
    count+=value;
}

void Sumator::reset() {
    count=0;
}

int Sumator::get_total() const {
    return count;
}

void Sumator::_bind_methods() {
    ObjectTypeDB::bind_method("add",&Sumator::add);
    ObjectTypeDB::bind_method("reset",&Sumator::reset);
    ObjectTypeDB::bind_method("get_total",&Sumator::get_total);
}

Sumator::Sumator() {
    count=0;
}
```

Then, the new class needs to be registered somehow, so two more files need to be created:

```
register_types.h
register_types.cpp
```

With the following contents

```
/* 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
}
```

Next, we need to create a SSub so the build system compiles this module:

```
# SSub
Import('env')

env.add_source_files(env.modules_sources, "*.cpp") # just add all cpp files to the build
```

And finally, the configuration file for the module, this is a simple python script that must be named ‘config.py’

```
# 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).

The second function allows to customize the build process for the module, like adding special compiler flags, options etc. (This can be done in SSub, but configure(env) is called at a previous stage). If unsure, just ignore this.

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/SSub
```

You can then zip it and share the module with everyone else. When building for every platform (instructions in the previous section), your module will be included.

## Using the module

Using your newly created module is very easy, from any script you can do:

```
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.

## 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 `[[API: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 `[[API: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.

### 13.1.6 Creating Android modules

#### Introduction

Making videogames 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 than you don't want even for your worst enemy.

## 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.

## 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 [[Compiling for Android|Android export templates]], as well as creating [[Custom modules in C++|custom modules]].

### 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:

```
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:

```
def can_build(plat):
    return plat=="android" or plat=="iphone"

def configure(env):
    if env['platform'] == 'android':
        #android specific code
```

## Java singleton

An android module will usually have a singleton class that will load it, this class inherits from Godot.SingletonBase. A singleton object template follows:

```
//namespace is wrong, will eventually change
package com.android.godot;

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 do
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() {}

protected void onGLDrawFrame(GL10 gl) {}
protected void onGLSurfaceChanged(GL10 gl, int width, int height) {} // singletons will always run in the main thread

}
```

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:

```
GodotLib.calldeferred("", "", new Object[]{param1, param2, etc});
```

Add this singleton to the build of the project by adding the following to `config.py`:

```
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
```

## 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:

```
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")
```

### 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:

```
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")
```

### 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 target sdk 15. If this ever changes, should be reflected in the manifest template:

<https://github.com/okamstudio/godot/blob/master/platform/android/AndroidManifest.xml.template>

Then, add the module folder to the project:

```
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", "")
```

### 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 build the ant project normally:

```
c:\\godot\\platform\\android\\java> ant release
```

This should generate the apk used as export template properly, as defined in the [[Compiling for Android|Android build instructions]].

Usually to generate the apk, again both commands must be run in sequence:

```
c:\\godot> scons p=android
c:\\godot\\platform\\android\\java> ant release
```

## Using the Module

To use the Module from GDScript, first enable the singleton by adding the following line to engine.cfg:

```
[android]
modules="com/android/godot/MySingleton"
```

More than one singleton module can be enable by separating with comma:

```
[android]
modules="com/android/godot/MySingleton, com/android/godot/MyOtherSingleton"
```

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") )
```

## 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.

## Future

Godot has an experimental Java API Wrapper that allows to use the entire Java API fro GDScript. It's simple to use and it's used like this:

```
class = JavaClassWrapper.wrap()
```

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](#).

## 13.2 Data and file formats

### 13.2.1 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 the [[API:File|File class]] as well as the packet APIs for [[API:PacketPeer]]. This 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:

Type	Value
0	null
1	bool
2	integer
3	float
4	string
5	vector2
6	rect2
7	vector3
8	matrix32
9	plane
10	quaternion
11	aabb (rect3)
12	matrix3x3
13	transform (matrix 4x3)
14	color
15	image
16	node path
17	rid (unsupported)
18	object (unsupported)
19	input event
20	dictionary
21	array
22	byte array
23	int array
24	float array
25	string array
26	vector2 array
27	vector3 array
28	color array

Following this is the actual packet contents, which varies for each type of packet:

### 0: null

### 1: bool

Offset	Len	Type	Description
4	4	Integer	0 for False, 1 for True

### 2: integer

Offset	Len	Type	Description
4	4	Integer	Signed, 32-Bit Integer

### 3: float

Offset	Len	Type	Description
4	4	Float	IEE 754 32-Bits Float

**4: string**

Offset	Len	Type	Description
4	4	Integer	String Length (in Bytes)
8	X	Bytes	UTF-8 Encoded String

This field is padded to 4 bytes.

**5: vector2**

Offset	Len	Type	Description
4	4	Float	X Coordinate
8	4	Float	Y Coordinate

**6: rect2**

Offset	Len	Type	Description
4	4	Float	X Coordinate
8	4	Float	Y Coordinate
12	4	Float	X Size
16	4	Float	Y Size

**7: vector3**

Offset	Len	Type	Description
4	4	Float	X Coordinate
8	4	Float	Y Coordinate
12	4	Float	Z Coordinate

**8: matrix32**

Offset	Len	Type	Description
4	4	Float	[0][0]
8	4	Float	[0][1]
12	4	Float	[1][0]
16	4	Float	[1][1]
20	4	Float	[2][0]
24	4	Float	[2][1]

**9: plane**

Offset	Len	Type	Description
4	4	Float	Normal X
8	4	Float	Normal Y
12	4	Float	Normal Z
16	4	Float	Distance

**10: quaternion**

Offset	Len	Type	Description
4	4	Float	Imaginary X
8	4	Float	Imaginary Y
12	4	Float	Imaginary Z
16	4	Float	Real W

**11: aabb (rect3)**

Offset	Len	Type	Description
4	4	Float	X Coordinate
8	4	Float	Y Coordinate
12	4	Float	Z Coordinate
16	4	Float	X Size
20	4	Float	Y Size
24	4	Float	Z Size

**12: matrix3x3**

Offset	Len	Type	Description
4	4	Float	[0][0]
8	4	Float	[0][1]
12	4	Float	[0][2]
16	4	Float	[1][0]
20	4	Float	[1][1]
24	4	Float	[1][2]
28	4	Float	[2][0]
32	4	Float	[2][1]
36	4	Float	[2][2]

**13: transform (matrix 4x3)**

Offset	Len	Type	Description
4	4	Float	[0][0]
8	4	Float	[0][1]
12	4	Float	[0][2]
16	4	Float	[1][0]
20	4	Float	[1][1]
24	4	Float	[1][2]
28	4	Float	[2][0]
32	4	Float	[2][1]
36	4	Float	[2][2]
40	4	Float	[3][0]
44	4	Float	[3][1]
48	4	Float	[3][2]

**14: color**

Offset	Len	Type	Description
4	4	Float	Red (0..1)
8	4	Float	Green (0..1)
12	4	Float	Blue (0..1)
16	4	Float	Alpha (0..1)

**15: image**

Offset	Len	Type	Description
4	4	Integer	Format (see FORMAT_* in “Image”:class_image)
8	4	Integer	Mip-Maps (0 means no mip-maps).
12	4	Integer	Width (Pixels)
16	4	Integer	Height (Pixels)
20	4	Integer	Data Length
24..24+DataLength	1	Byte	Image Data

This field is padded to 4 bytes.

**16: node path**

Off-set	Len	Type	Description
4	4	Integer	String Length, or New Format (val&gt;0x80000000!=0 and NameCount=val&gt;0x7FFFFFFF)

For old format:	Offset	Len	Type	Description
	8	X	Bytes	UTF-8 Encoded String

Padded to 4 bytes.

For new format:	Offset	Len	Type	Description
	4	4	Integer	Sub-Name Count
	8	4	Integer	Flags (absolute: val&gt;1 != 0 )

For each Name and Sub-Name

Offset	Len	Type	Description
X+0	4	Integer	String Length
X+4	X	Bytes	UTF-8 Encoded String

Every name string is is padded to 4 bytes.

**17: rid (unsupported)****18: object (unsupported)****19: input event****20: dictionary**

Offset	Len	Type	Description
4	4	Integer	val&amp;#0x7FFFFFFF = elements, val&amp;#0x80000000 = shared (bool)

Then what follows is, for amount of “elements”, pairs of key and value, one after the other, using this same format.

**21: array**

Offset	Len	Type	Description
4	4	Integer	val&amp;#0x7FFFFFFF = elements, val&amp;#0x80000000 = shared (bool)

Then what follows is, for amount of “elements”, values one after the other, using this same format.

**22: byte array**

Offset	Len	Type	Description
4	4	Integer	Array Length (Bytes)
8..8+length	1	Byte	Byte (0..255)

The array data is padded to 4 bytes.

**23: int array**

Offset | Len | Type | Description  
 - | - | - | -  
 4|4|Integer| Array Length (Integers)  
 8..8+length\*4|4|Integer| 32 Bits Signed Integer

**24: float array**

Offset | Len | Type | Description  
 - | - | - | -  
 4|4|Integer| Array Length (Floats)  
 8..8+length\*4|4|Integer| 32 Bits IEE 754 Float

**25: string array**

Offset	Len	Type	Description
4	4	Integer	Array Length (Strings)

For each String:

Offset	Len	Type	Description
X+0	4	Integer	String Length
X+4	X	Bytes	UTF-8 Encoded String

Every string is padded to 4 bytes.

**26: vector2 array**

Offset	Len	Type	Description
4	4	Integer	Array Length
8..8+length_8 4 Float  X Coordinate	4	Float	Y Coordinate

**27: vector3 array**

Offset | Len | Type | Description  
- | - | - | -  
4|4|Integer| Array Length  
8..8+length\_12|4|Float| X Coordinate  
8..12+length\_12|4|Float| Y Coordinate  
8..16+length\*12|4|Float| Z Coordinate

**28: color array**

Offset	Len	Type	Description
4	4	Integer	Array Length
8..8+length_16 4 Float  Red (0..1)	4	Float	Green (0..1)
8..16+length_16 4 Float  Blue (0..1)	4	Float	Alpha (0..1)

## 13.3 Misc

### 13.3.1 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.

#### 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`.

## 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.

## 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.

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

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
```

## 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
```

## 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
```

## 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
```

## 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.

### Running a script

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).

### 13.3.2 Changing editor fonts

Godot allows changing the font for the editor, and the font for the code editor. Both need to be in .fnt format, so they need to be imported somewhere using the [[Importing fonts|font import tool]].

Then copy or do whatever you want with the font, as long as the location does not change, and set the relevant property in Editor Settings. Code editor font is refreshed automatically, but the editor needs to be restarted for the new global font to take effect.

### 13.3.3 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:

```
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:

```
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.get_pending_event_count() > 0:
        var event = InAppStore.pop_pending_event()
        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:

- `get_pending_event_count()` Returns the number of pending events on the queue.
- `Variant pop_pending_event()` 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

```
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:

```
var result = InAppStore.purchase( { "product_id": "my_product" } )
```

**Response event** The response event will be a dictionary with the following fields:

On error:

```
{  
    "type": "purchase",  
    "result": "error",  
    "product_id": "the product id requested"  
}
```

On success:

```
{  
    "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:

```
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:

```
{  
    "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:

```
var result = GameCenter.post_score( { "value": 100, "category": "my_leaderboard", } )
```

**Response event** The response event will be a dictionary with the following fields:

On error:

```
{
  "type": "post_score",
  "result": "error",
  "error_code": the value from NSError::code,
  "error_description": the value from NSError::localizedDescription,
}
```

On success:

```
{
  "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:

```
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:

```
{  
    "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 ],
  "progress": [ list of the progress made on 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:

```
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:

```
{  
    "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:

```
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
```

---

## Contributing

---

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.

### 14.1 Issues management

GitHub proposes three features to manage issues:

- Set one or several labels from a predefined list
- Set one milestone from a predefined list
- 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 and we are not sure yet to what extent we will use it or OpenProject instead, we will not use assignees extensively for the time being.

#### 14.1.1 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.
- *Enhancement*: describes a proposed enhancement to an existing functionality.
- *Feature request*: describes a wish for a new feature to be implemented.
- *High priority*: the issue should be treated in priority (typically critical bugs).
- *Needs discussion*: the issue is not consensual and needs further discussion to define what exactly should be done to address the topic.

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 *Bug*, *Confirmed* and *High priority* at the same time if it's a critical bug that was confirmed by several users, or *Feature request* and *Needs discussion* if it's a non-consensual feature request, or one that is not precise enough to be worked on.

**Topics:**

- *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.
- *Demos*: relates to the official demos.
- *GDScript*: relates to GDScript.
- *Porting*: relates to some specific platforms.
- *Rendering engine*: relates to the 2D and 3D rendering engines.
- *User interface*: relates to the UI design.

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 topic.

Bug reports concerning the website or the documentation should not be filed in GitHub but in the appropriate tool in OpenProject, therefore such issues should be closed and archived once they have been moved to their rightful platform.

**Platforms:** *Android, HTML5, iOS, Linux, OS X, Windows*

By default, it is assumed that a given issue applies to all platforms. If one of the platform labels is used, it is the 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).

### 14.1.2 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 set to *Later*. 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 :)

## 14.2 Documentation writing and translating guidelines

This page describes the rules to follow if you want to contribute Godot Engine by writing documentation or translating existing documentation.

### 14.2.1 What is a good documentation ?

A good documentation is well written in plain English and well-formed sentences. It is clear and objective.

A documentation page is not a tutorial page. We differentiate these concepts by these definitions :

- **tutorial** : a page aiming at explaining how to use one or more concepts in Godot Editor in order to achieve a specific goal with a learning purpose (ie. “make a simple 2d Pong game”, “apply forces to an object”...)

- documentation : a page describing precisely one and only one concept at the time, if possible exhaustively (ie. the list of methods of the Sprite class for example).

You are free to write the kind of documentation you wish, as long as you respect the following rules.

### 14.2.2 Create a new wiki page

Creating a new documentation page or tutorial page is easy. The following rules must be respected :

- Choose a short and explicit title
- Respect the grammar and orthography
- Make use of the [[Wiki syntax]]

Try to structure your page in order to enable users to include a page directly in another page or even forum posts using the include wiki syntax. For example, the syntax to include the page you are reading is :

```
!{{include(Documentation writing and translating guidelines)}}.
```

#### Titles

Please always begin pages with their name:

```
h1. <Insert your title here>
```

Also, avoid American CamelCase titles: titles' 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 names (and people names) should have capitalized first letter. This is good:

- Starting up with Godot Engine and this is bad:
- Starting up with godot engine

### 14.2.3 Note for non-English authors

If you intend to create a new page in your language, you are asked to firstly create the corresponding English page if it doesn't already exist. **Do it even if you will not write it yourself, just leave it blank.** Only then, create the corresponding page in your own language. Maybe later, another contributor will translate your new page to English.

**Remember :** even if Godot aims at being accessible to everyone, English is the most frequent language for documentation.

### 14.2.4 Translating existing pages

You are very welcome to translate existing pages from English to your language, or from your language to English. If these guidelines were respected, an English page already exists for every page of this wiki, even if it is empty. To translate an existing page, please follow these few rules :

- Respect the grammar and orthography

- Make use of the [[wiki syntax]]
- Re-use images
- Always keep the structure of the English page (if it is written yet, follow the structure of the original language page you are translating from).

To translate an existing page, simply copy its original content. Then, create the new page in the section of your language, copy the English content in it and start translating.

Please add a line at the very beginning of your translation, linking to the English base page you translate from :  
Traduction de ![[Godot Engine:Creating 2D Games]]

The previous link is of the form ![:] which enables you to add a link to a page located in an other project. Here, “Godot Engine” is the English project.

#### **14.2.5 Important changes and discussions**

You are welcome to correct mistakes or styles to respect these guidelines. However, in case of important changes, please do not start a discussion on this page : use the forum, create a new topic with a link to the incriminated page and start discussing there about your remarks.

#### **14.2.6 Licence**

This wiki and every page it contains is published under the terms of the Creative Commons BY 3.0 license.

### **14.3 List of classes and documenters**

Status list : Not started, Started, Finished, Removed

Class name   Assigned to   Status   Start date   Notes
@GDScript
@Global Scope
AABB   bojidar_bg   Finished
AcceptDialog
AnimatedSprite
AnimatedSprite3D
Animation
AnimationPlayer
AnimationTreePlayer
Area
Area2D   Ovnuniarchos   Finished   2015/12/22
Array   vnen   Started   10/10/2015
AtlasTexture
AudioServer   Akien   Finished
AudioServerSW   Akien   Finished

AudioStream	Akien	Finished		
AudioStreamMPC	Akien	Finished		
AudioStreamOGGVorbis	Akien	Finished		
AudioStreamPlayback	Akien	Finished		
AudioStreamSpeex	Akien	Finished		
BackBufferCopy				
BakedLight				
BakedLightInstance				
BakedLightSampler				
BaseButton				
BitMap				
BoneAttachment				
BoxContainer				
BoxShape				
Button				
ButtonArray				
ButtonGroup				
Camera				
Camera2D				
CanvasItem				
CanvasItemMaterial				
CanvasItemShader				
CanvasItemShaderGraph				
CanvasLayer				
CanvasModulate				
CapsuleShape				
CapsuleShape2D	Ovnuniarchos	Finished		
CenterContainer				
CheckBox				
CheckButton				
CircleShape2D	Ovnuniarchos	Finished		
CollisionObject				
CollisionObject2D	Ovnuniarchos	Finished	2015/12/22	
CollisionPolygon				
CollisionPolygon2D	Ovnuniarchos	Finished		
CollisionShape				
CollisionShape2D	Ovnuniarchos	Finished		
Color				
ColorArray				
ColorPicker				
ColorPickerButton				
ColorRamp				
ConcavePolygonShape				
ConcavePolygonShape2D	Ovnuniarchos	Finished		
ConeTwistJoint				
ConfigFile				
ConfirmationDialog				

Container				
Control				
ConvexPolygonShape				
ConvexPolygonShape2D	Ovnuniarchos	Finished		
CubeMap				
Curve2D	Ovnuniarchos	Finished		
Curve3D	Ovnuniarchos	Finished		
DampedSpringJoint2D				
Dictionary				
DirectionalLight				
Directory	vnen	Started	11/10/2015	
EditorFileDialog				
EditorImportPlugin				
EditorPlugin				
EditorScenePostImport				
EditorScript				
Environment				
EventPlayer				
EventStream				
EventStreamChibi				
File				
FileDialog				
FixedMaterial				
Font				
FuncRef				
GDFunctionState				
GDNativeClass				
GDScript				
Generic6DOFJoint				
Geometry				
GeometryInstance				
Globals				
GraphEdit	StraToN	Finished		may need a tutorial. I'll think about it.
GraphNode	StraToN	Finished		may need a tutorial. I'll think about it.
GridContainer				
GridMap				
GrooveJoint2D				
HBoxContainer				
HButtonArray				
HScrollBar				
HSeparator				
HSlider				
HSplitContainer				
HTTPClient				
HingeJoint				
IP				
IP\_Unix				

Image				
ImageTexture				
ImmediateGeometry				
Input				
InputDefault				
InputEvent				
InputEventAction				
InputEventJoyButton				
InputEventJoyMotion				
InputEventKey				
InputEventMouseButton				
InputEventMouseMotion				
InputEventScreenDrag				
InputEventScreenTouch				
InputMap				
IntArray				
InterpolatedCamera				
ItemList				
Joint				
Joint2D				
KinematicBody				
KinematicBody2D	Ovnuniarchos	Started	2015/11/23	
Label				
LargeTexture				
Light				
Light2D				
LightOccluder2D				
LineEdit				
LineShape2D	Ovnuniarchos	Finished		
MainLoop				
MarginContainer				
Marshalls				
Material				
MaterialShader				
MaterialShaderGraph				
Matrix3				
Matrix32				
MenuButton				
Mesh				
MeshDataTool				
MeshInstance				
MeshLibrary				
MultiMesh				
MultiMeshInstance				
Mutex				
Navigation				
Navigation2D				

NavigationMesh					
NavigationMeshInstance					
NavigationPolygon					
NavigationPolygonInstance					
Nil					
Node					
Node2D					
NodePath					
OS					
Object					
OccluderPolygon2D					
OmniLight					
OptionButton					
PCKPacker					
PHashTranslation					
PackedDataContainer					
PackedDataContainerRef					
PackedScene					
PacketPeer					
PacketPeerStream					
PacketPeerUDP					
Panel					
PanelContainer					
ParallaxBackground					
ParallaxLayer					
ParticleAttractor2D					
Particles					
Particles2D					
Patch9Frame					
Path	Ovnuniarchos	Finished			
Path2D	Ovnuniarchos	Finished			
PathFollow	Ovnuniarchos	Finished			
PathFollow2D	Ovnuniarchos	Finished			
PathRemap					
Performance					
Physics2DDirectBodyState					
Physics2DDirectBodyStateSW					
Physics2DDirectSpaceState					
Physics2DServer					
Physics2DServerSW					
Physics2DShapeQueryParameters					
Physics2DShapeQueryResult					
Physics2DTestMotionResult					
PhysicsBody					
PhysicsBody2D	Ovnuniarchos	Finished	2015/12/22		
PhysicsDirectBodyState					
PhysicsDirectBodyStateSW					

PhysicsDirectSpaceState				
PhysicsServer				
PhysicsServerSW				
PhysicsShapeQueryParameters				
PhysicsShapeQueryResult				
PinJoint				
PinJoint2D				
Plane				
PlaneShape				
Polygon2D				
PolygonPathFinder				
Popup				
PopupDialog				
PopupMenu				
PopupPanel				
Portal				
Position2D				
Position3D				
ProgressBar				
ProximityGroup				
Quad				
Quat				
RID				
Range				
RawArray				
RayCast				
RayCast2D	eska	Started	2015-10-16	
RayShape				
RayShape2D	Ovnuniarchos	Finished		
RealArray				
Rect2	bojidar\_bg	Finished		
RectangleShape2D	Ovnuniarchos	Finished		
Reference				
ReferenceFrame				
RegEx	Ovnuniarchos	Finished	2015-11-03	
RemoteTransform2D	eska	Started	2015-10-16	
RenderTargetTexture				
Resource				
ResourceImportMetadata				
ResourceInteractiveLoader				
ResourceLoader				
ResourcePreloader				
ResourceSaver				
RichTextLabel				
RigidBody				
RigidBody2D	Ovnuniarchos	Started	2015/11/23	
Room				

RoomBounds				
Sample	Akien	Finished		
SampleLibrary	Akien	Finished		
SamplePlayer	Akien	Finished		
SamplePlayer2D	Akien	Finished		
SceneTree				
Script				
ScrollBar				
ScrollContainer				
SegmentShape2D	Ovnuniarchos	Finished		
Semaphore				
Separator				
Shader				
ShaderGraph				
ShaderMaterial				
Shape				
Shape2D	Ovnuniarchos	Finished		
Skeleton				
Slider				
SliderJoint				
SoundPlayer2D	Akien	Not started		
SoundRoomParams	Akien	Not started		
Spatial	Akien	Not started		
SpatialPlayer	Akien	Not started		
SpatialSamplePlayer	Akien	Not started		
SpatialSound2DServer	Akien	Not started		
SpatialSound2DServerSW	Akien	Not started		
SpatialSoundServer	Akien	Not started		
SpatialSoundServerSW	Akien	Not started		
SpatialStreamPlayer	Akien	Not started		
SphereShape				
SpinBox				
SplitContainer				
SpotLight				
Sprite				
Sprite3D				
SpriteBase3D				
SpriteFrames				
StaticBody				
StaticBody2D	Ovnuniarchos	Started	2015/11/23	
StreamPeer				
StreamPeerSSL				
StreamPeerTCP				
StreamPlayer				
String				
StringArray				
StyleBox				

StyleBoxEmpty		
StyleBoxFlat		
StyleBoxImageMask		
StyleBoxTexture		
SurfaceTool		
TCP\_Server		
TabContainer		
Tabs		
TestCube		
TextEdit		
Texture		
TextureButton		
TextureFrame		
TextureProgress		
Theme		
Thread		
TileMap	Akien	Finished
TileSet	Akien	Finished
Timer	Akien	Finished
ToolButton		
TouchScreenButton		
Transform		
Translation		
TranslationServer		
Tree		
TreeItem		
Tween		
UndoRedo		
VBoxContainer		
VButtonArray		
VScrollBar		
VSeparator		
VSlider		
VSplitContainer		
Vector2	bojidar\_bg	Finished
Vector2Array	bojidar\_bg	Finished
Vector3	bojidar\_bg	Finished
Vector3Array	bojidar\_bg	Finished
VehicleBody		
VehicleWheel		
VideoPlayer		
VideoStream		
Viewport		
ViewportSprite		
VisibilityEnabler		
VisibilityEnabler2D		
VisibilityNotifier		

VisibilityNotifier2D			
VisualInstance			
VisualServer			
WeakRef			
WindowDialog			
World			
World2D			
WorldEnvironment			
XMLParser			
YSort	eska	Started	2015-10-16
bool			
float			
int			

## 14.4 Reference filling work

Godot Engine provides an important number of classes that you can make use of to create your games. However, the [[Reference|reference]] that lists all these classes with their methods is quite incomplete. We need your kind help to fill this reference. This page will explain you how.

> Please note: we aim at filling completely this reference in English first. Please do not start translating it for the moment.

[[List of classes and documenters]]

### 14.4.1 Editing with Github

#### 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.

```
git clone https://github.com/your_name/godot.git
```

Then, create a new git branch that will contain your changes.

```
git checkout -b reference-edition
```

The branch you just created is identical to current master branch of Godot Engine. It already contains a doc/ folder, with the currently written reference.

#### 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 Godot 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 (or not) 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.

### Push and request a pull of your changes

Once your modifications are finished, push your changes on your GitHub repository:

```
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 on the GitHub UI of your Godot fork.

## 14.4.2 Edit doc/base/classes.xml file

First of all, check the [[List of classes and documenters]]. Try to work on classes not already assigned nor filled.

This file is produced by Godot Engine. It is used by the editor, for example in the Help window (F1, Shift+F1). You can edit this file using your favourite text editor.

Here is an example with the Node2D class:

Base node **for** 2D system.

Base node **for** 2D system. Node2D contains a position, rotation **and** scale, which **is** used to position an

Set the position of the 2d node.

Set the rotation of the 2d node.

Set the scale of the 2d node.

```
Return the position of the 2D node.
```

```
Return the rotation of the 2D node.
```

```
Return the scale of the 2D node.
```

```
Return the global position of the 2D node.
```

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 the brief\_description of the class, but in our case they are already filled. Let's edit the description of the rotate() method:

```
Rotates the node of "degrees" degrees.
```

That's all!

You simply have to write any missing text between these marks:

- 
- 
- 

Describe clearly and shortly what it does. You can include an example of use if needed. Avoid grammar faults.

#### 14.4.3 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 [Forums](#) and on IRC (freenode, #godotengine).

### 14.5 Wiki Syntax

This page is a helper for Wiki syntax. TODO.