Sign in here!

fsu.devlup.org/signin

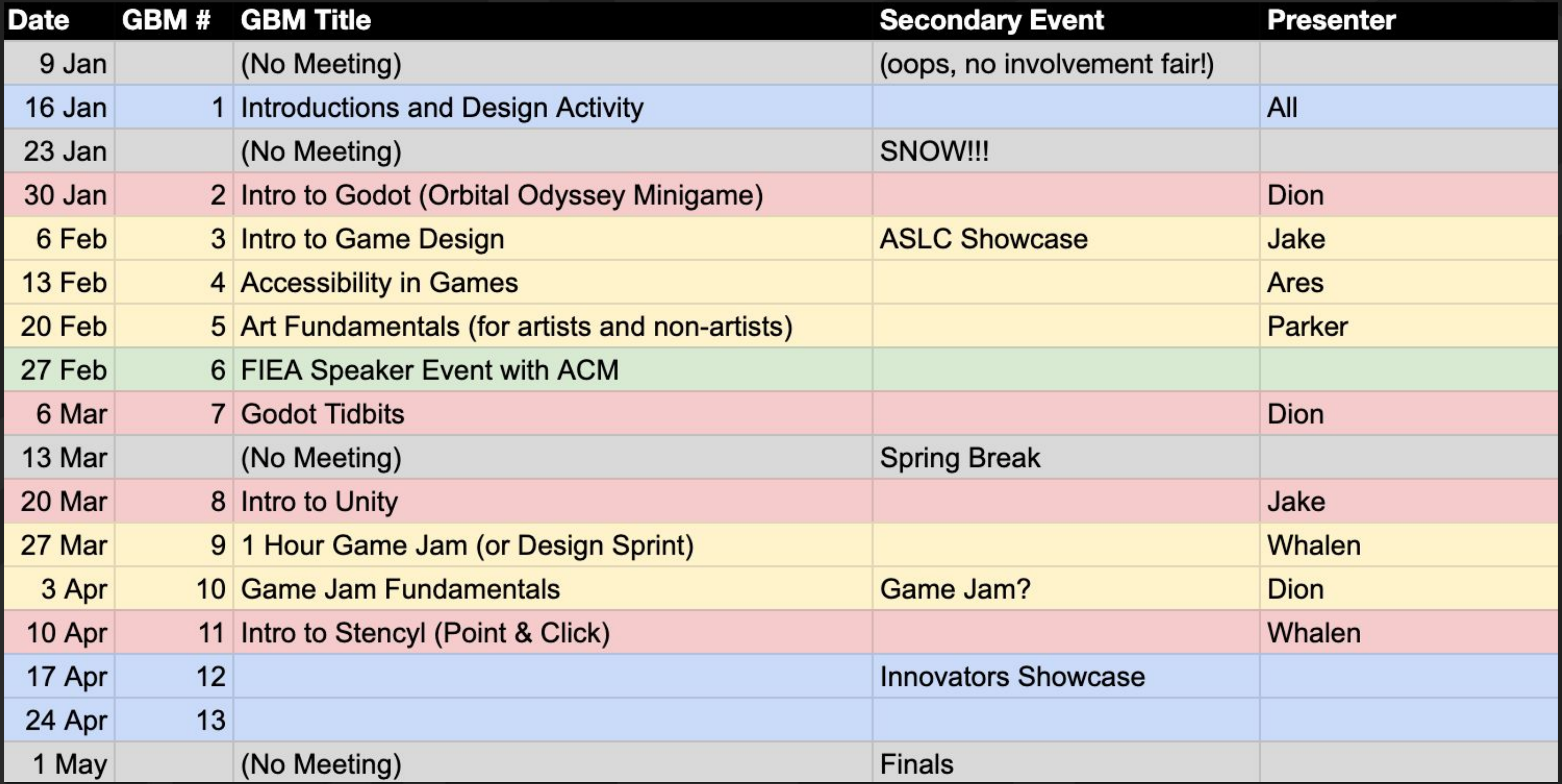
DevLUp FSU

GBM #6

Godot Tidbits

March 6th, 2025

Meeting Schedule



Date	GBM #	GBM Title	Secondary Event	Presenter
9 Jan		(No Meeting)	(oops, no involvement fair!)	
16 Jan	1	Introductions and Design Activity		All
23 Jan		(No Meeting)	SNOW!!!	
30 Jan	2	Intro to Godot (Orbital Odyssey Minigame)		Dion
6 Feb	3	Intro to Game Design	ASLC Showcase	Jake
13 Feb	4	Accessibility in Games		Ares
20 Feb	5	Art Fundamentals (for artists and non-artists)		Parker
27 Feb	6	FIEA Speaker Event with ACM		
6 Mar	7	Godot Tidbits		Dion
13 Mar		(No Meeting)	Spring Break	
20 Mar	8	Intro to Unity		Jake
27 Mar	9	1 Hour Game Jam (or Design Sprint)		Whalen
3 Apr	10	Game Jam Fundamentals	Game Jam?	Dion
10 Apr	11	Intro to Stencyl (Point & Click)		Whalen
17 Apr	12		Innovators Showcase	
24 Apr	13			
1 May		(No Meeting)	Finals	

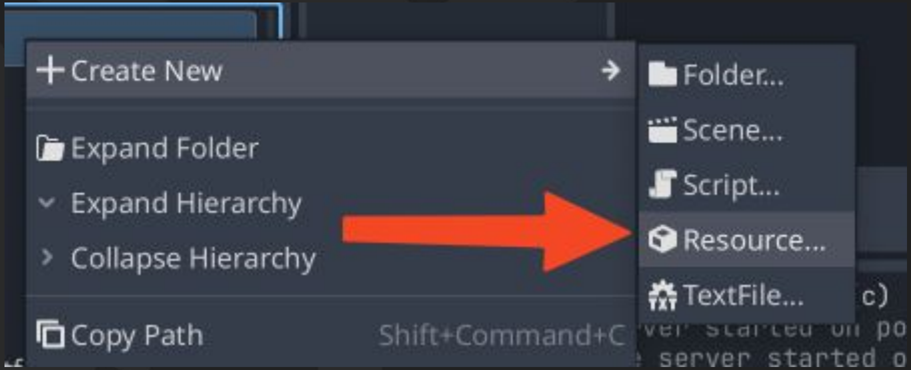
Officer Interest Form for Next Year

- President
- Vice President
- Treasurer
- Secretary
- Marketing Chair
- Social Chair
- Creative Chair

#👀showoff

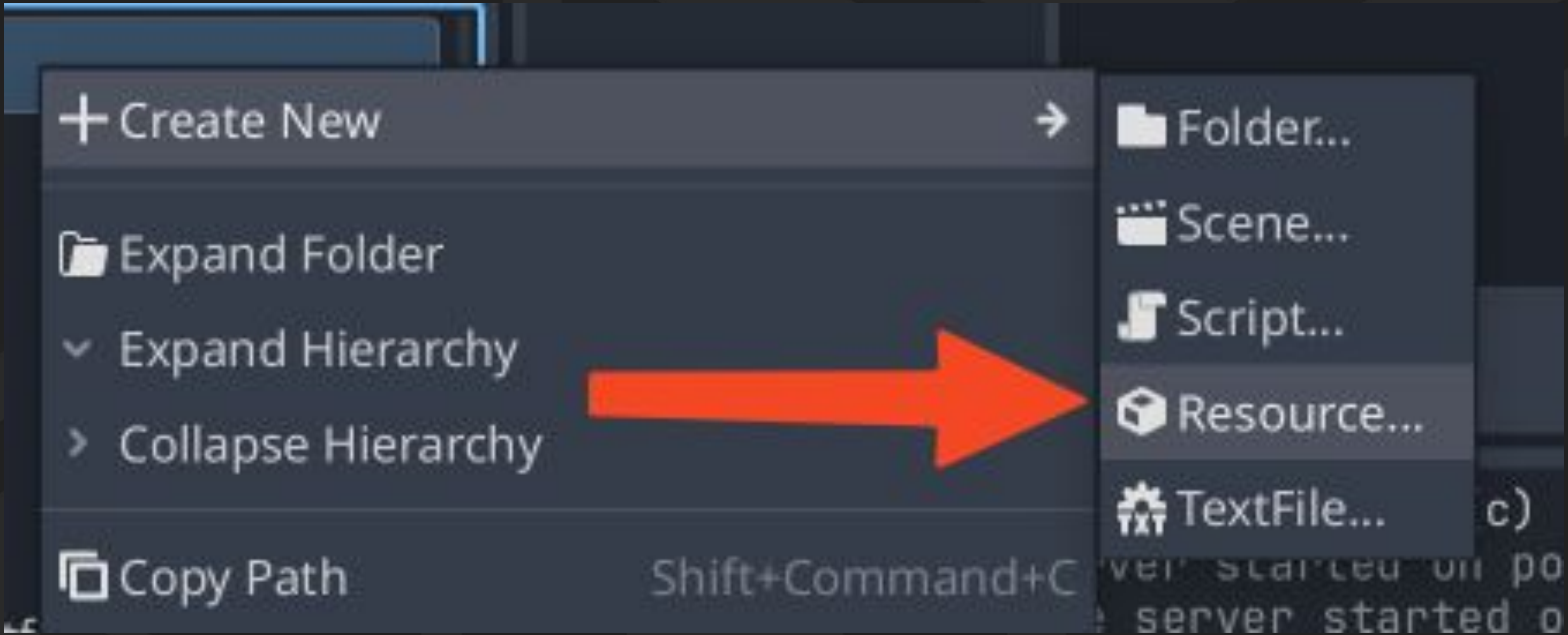
Topics We'll Cover

- Resources
- Autoloads
- Signals
- Plugins
- Importing Assets
- Using C#
- What's New in Godot 4.4!



Resources

What is a resource?



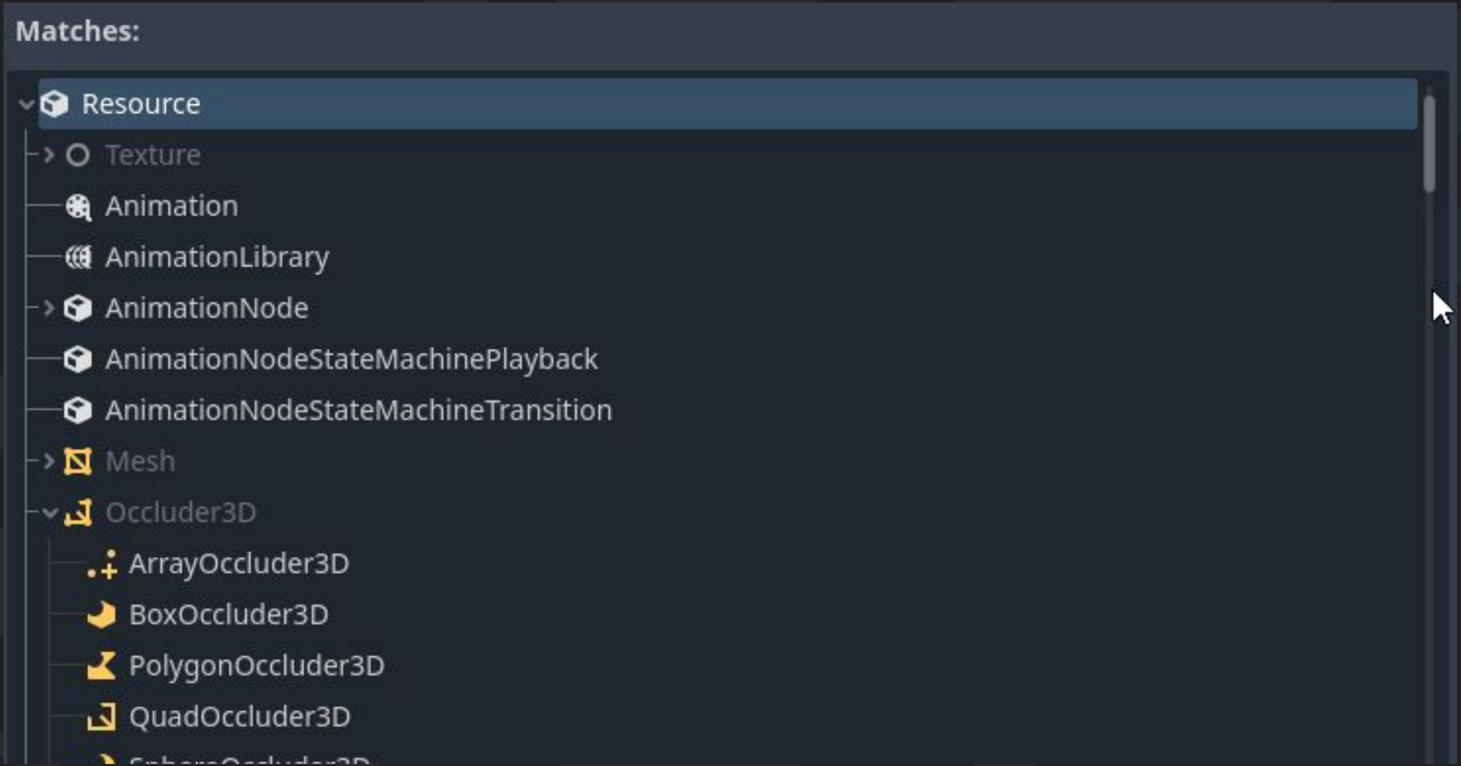
What is a resource?

- A resource is a **data container**.
- A node *does* store data, but mostly gives *functionality*.
  - E.g., draws sprites, renders 3D models, simulates physics, arranges UI elements
- A resource can’t do anything by itself, but is used by nodes.

- Everything that Godot saves to a file is a resource.
  - Images, scripts, and even scenes are resources!

- More or less equivalent to Unity’s ScriptableObject.

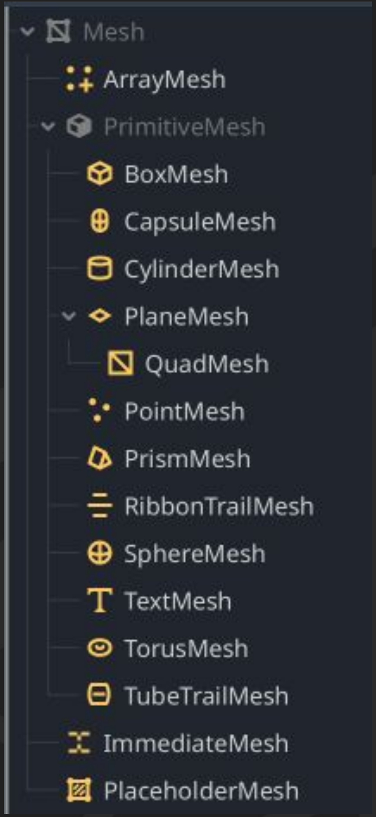
You’ve already used resources



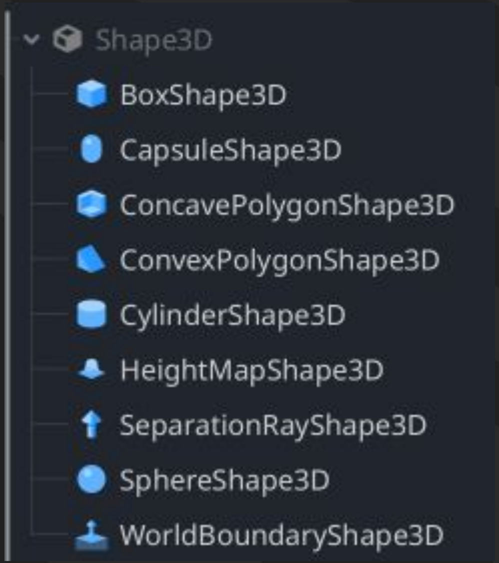
Matches:

- Resource
  - Texture
  - Animation
  - AnimationLibrary
  - AnimationNode
  - AnimationNodeStateMachinePlayback
  - AnimationNodeStateMachineTransition
  - Mesh
  - Occluder3D
    - ArrayOccluder3D
    - BoxOccluder3D
    - PolygonOccluder3D
    - QuadOccluder3D

You’ve already used resources



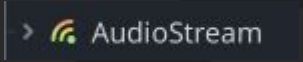
- Mesh
  - ArrayMesh
  - PrimitiveMesh
    - BoxMesh
    - CapsuleMesh
    - CylinderMesh
    - PlaneMesh
      - QuadMesh
    - PointMesh
    - PrismMesh
    - RibbonTrailMesh
    - SphereMesh
    - TextMesh
    - TorusMesh
    - TubeTrailMesh
  - ImmediateMesh
  - PlaceholderMesh



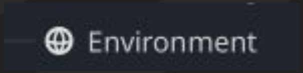
- Shape3D
  - BoxShape3D
  - CapsuleShape3D
  - ConcavePolygonShape3D
  - ConvexPolygonShape3D
  - CylinderShape3D
  - HeightMapShape3D
  - SeparationRayShape3D
  - SphereShape3D
  - WorldBoundaryShape3D

StandardMaterial3D

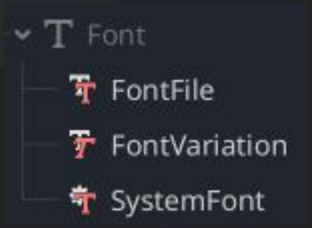
- Animation
- AnimationLibrary



- AudioStream

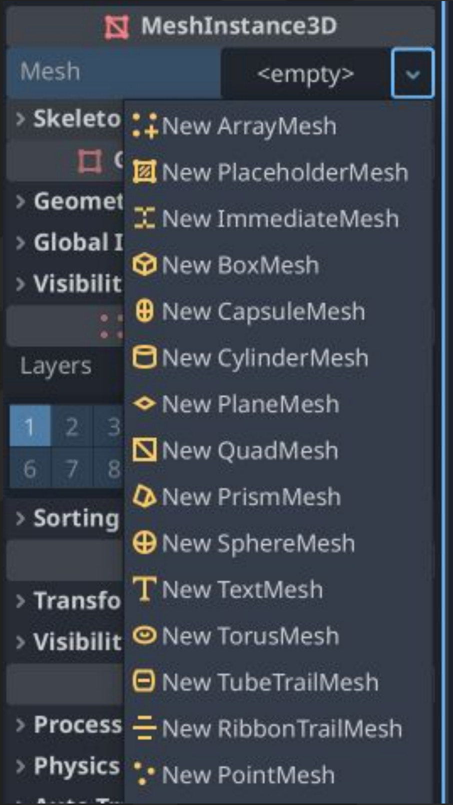


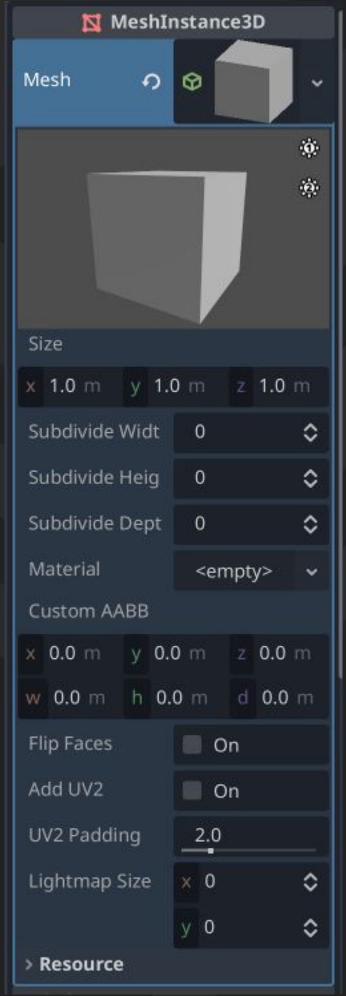
- Environment



- Font
  - FontFile
  - FontVariation
  - SystemFont

How nodes use resources

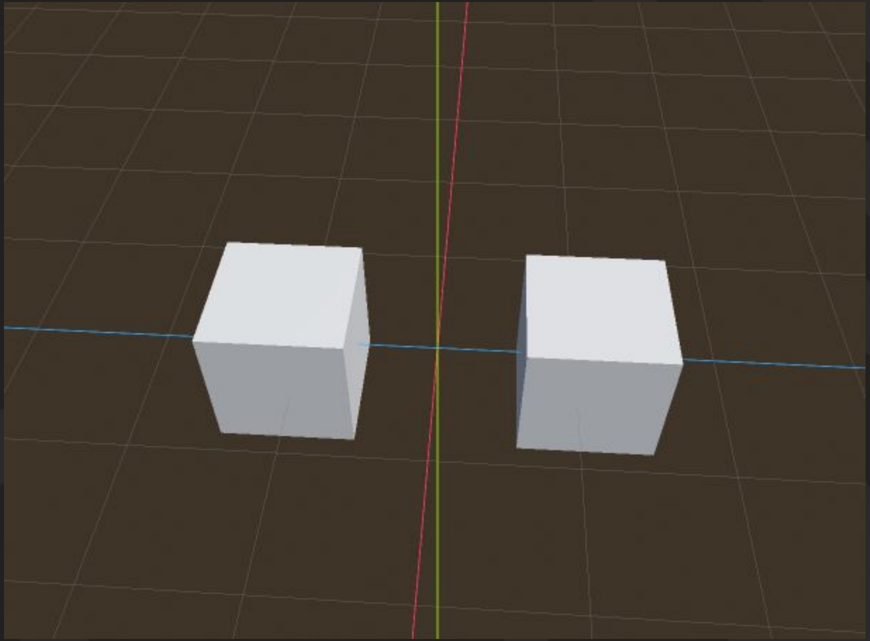


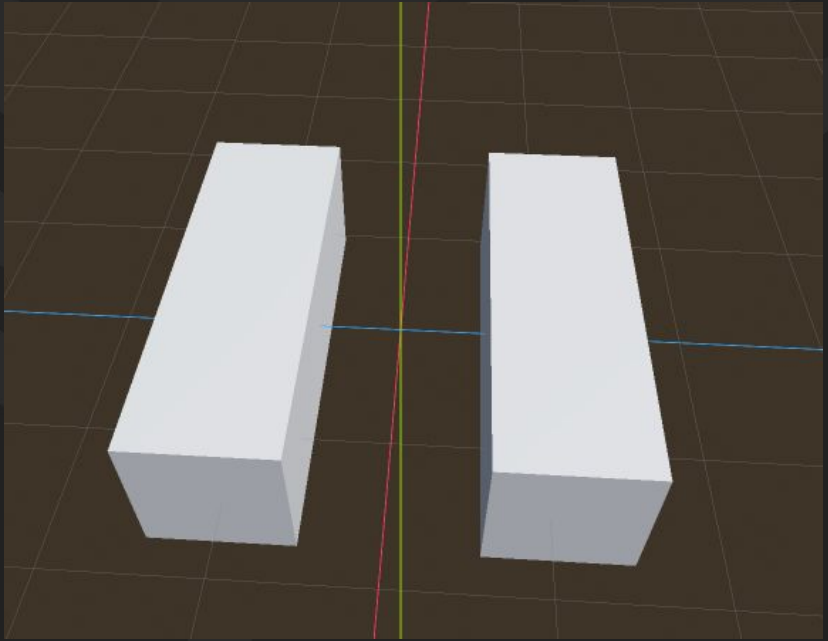

MeshInstance3D

Mesh

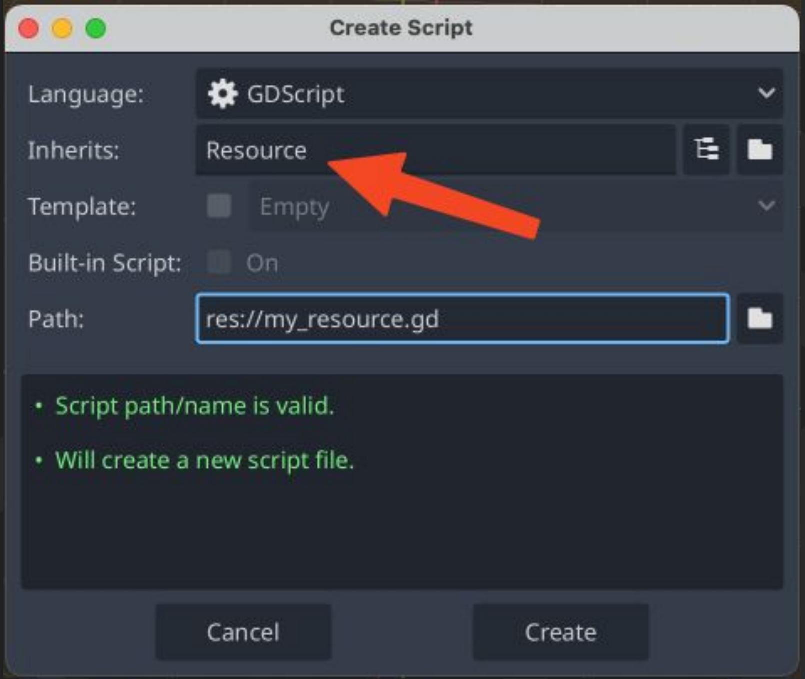
Size: x 1.0 m, y 1.0 m, z 1.0 m
Subdivide Widt: 0
Subdivide Heig: 0
Subdivide Dept: 0
Material: <empty>
Custom AABB: x 0.0 m, y 0.0 m, z 0.0 m, w 0.0 m, h 0.0 m, d 0.0 m
Flip Faces: On
Add UV2: On
UV2 Padding: 2.0
Lightmap Size: x 0, y 0
Resource

Resources are shared





Custom resources!



Create Script

Language: GDScript

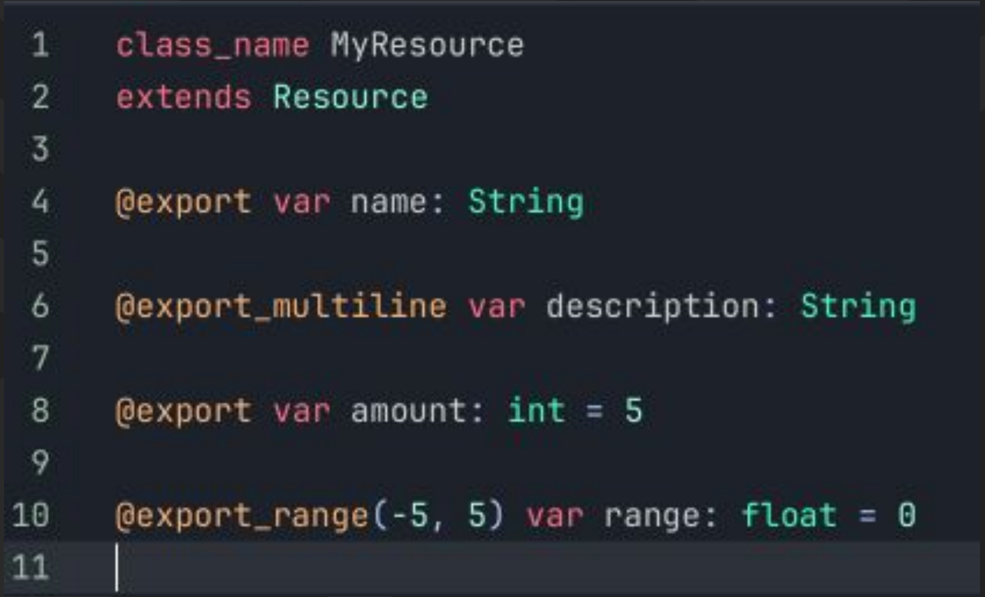
Inherits: Resource

Template: Empty

Built-in Script: On

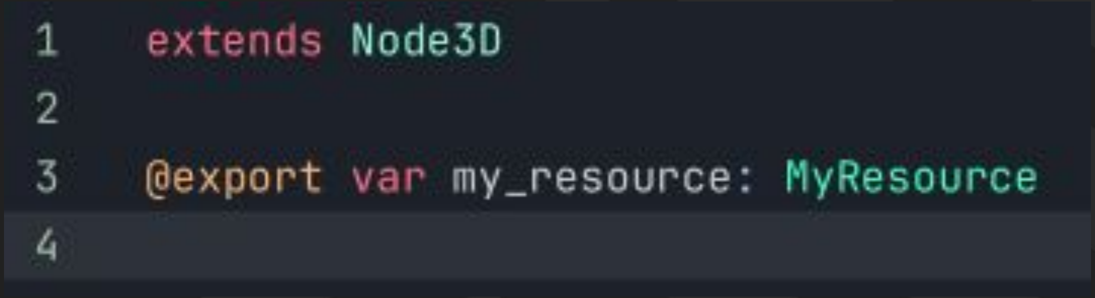
Path: res://my_resource.gd

- Script path/name is valid.
- Will create a new script file.

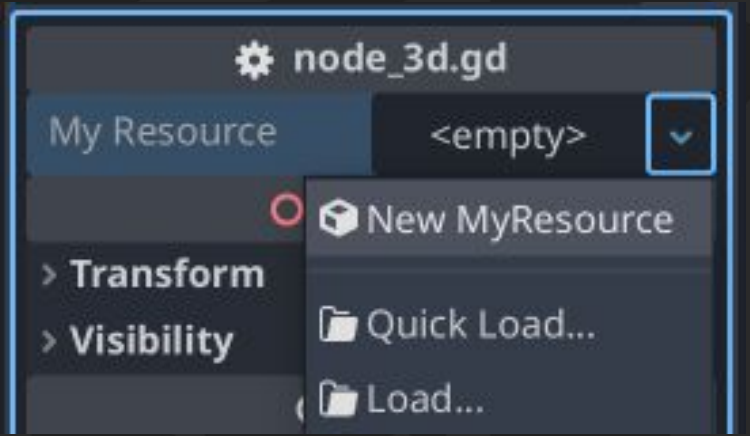


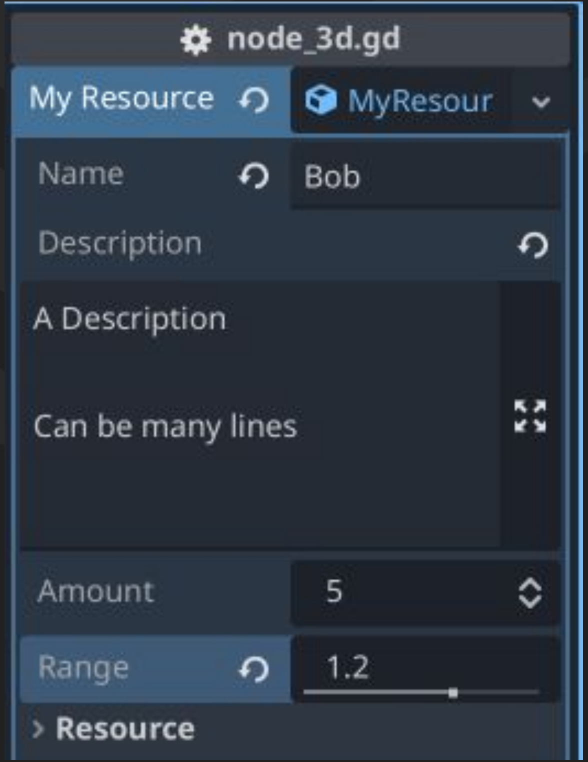
```
class_name MyResource
extends Resource

@export var name: String

@export_multiline var description: String

@export var amount: int = 5

@export_range(-5, 5) var range: float = 0
```

Custom resources!

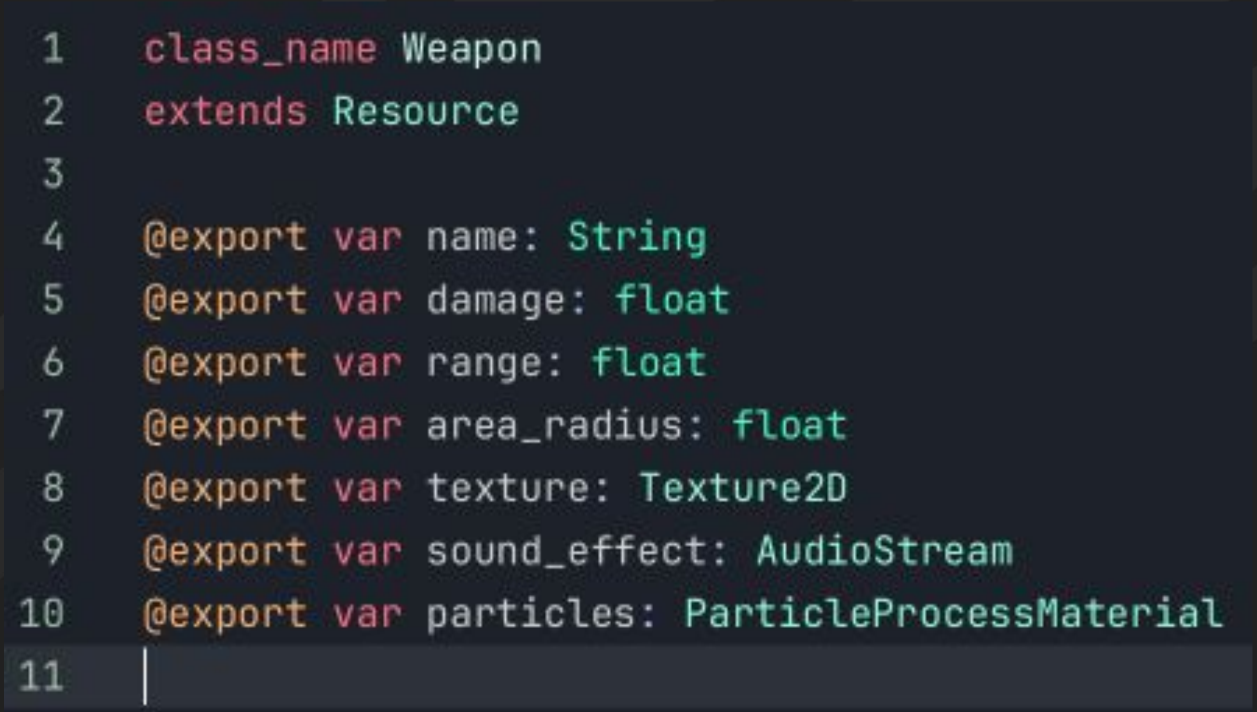


```
extends Node3D

@export var my_resource: MyResource

```



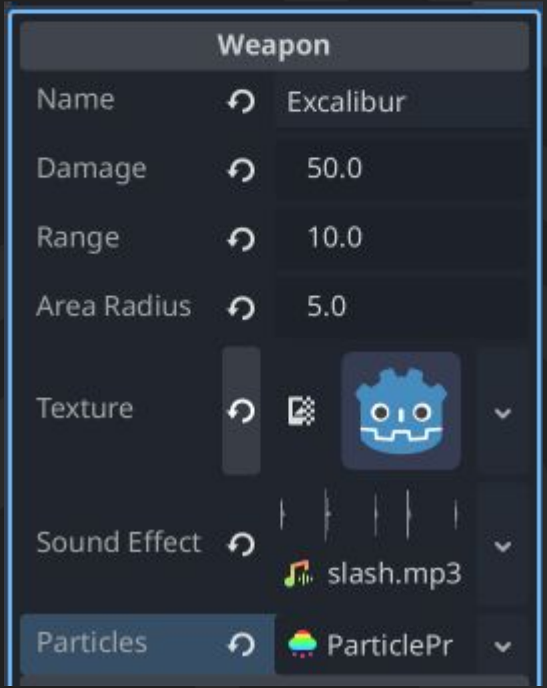


An example custom resource



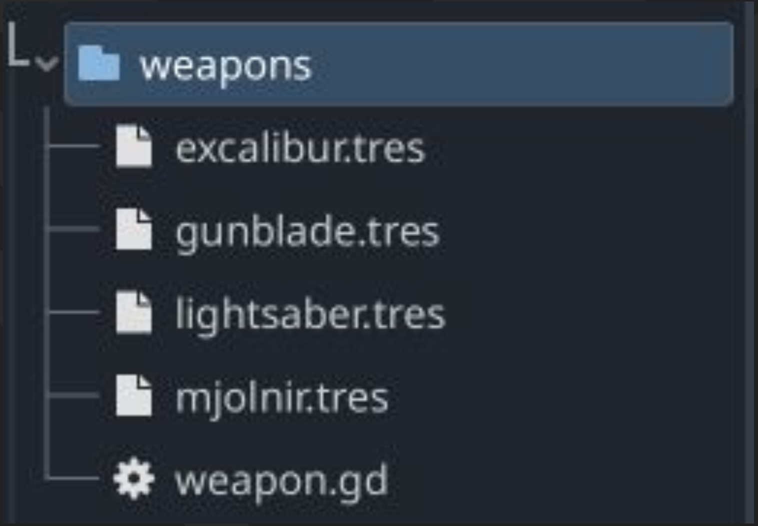
```
class_name Weapon
extends Resource

@export var name: String
@export var damage: float
@export var range: float
@export var area_radius: float
@export var texture: Texture2D
@export var sound_effect: AudioStream
@export var particles: ParticleProcessMaterial

```

An example custom resource



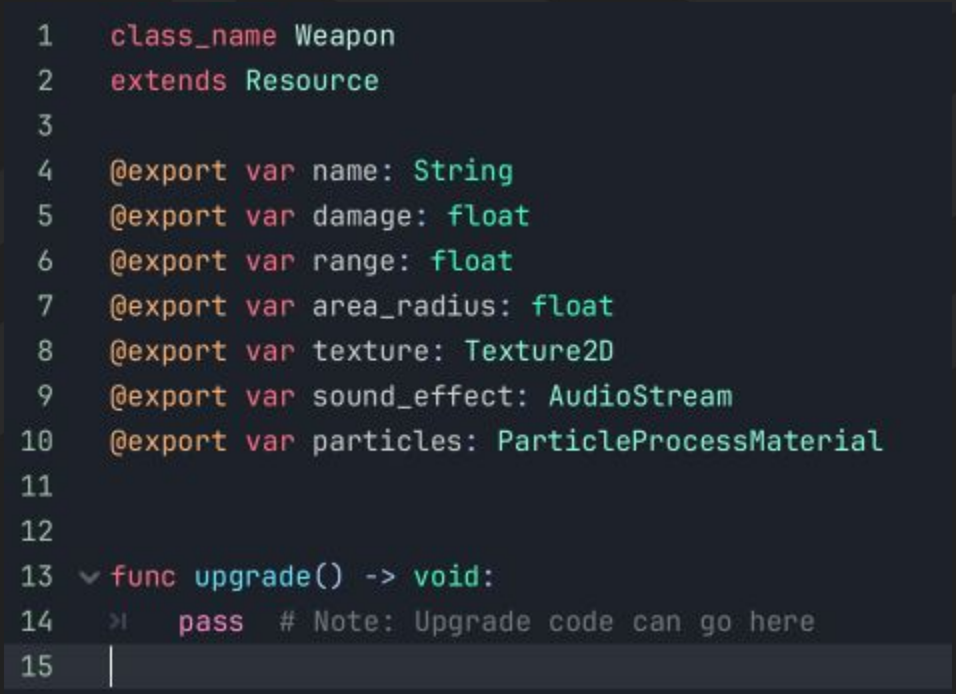
Weapon

Name	Excalibur
Damage	50.0
Range	10.0
Area Radius	5.0
Texture	
Sound Effect	slash.mp3
Particles	ParticlePr



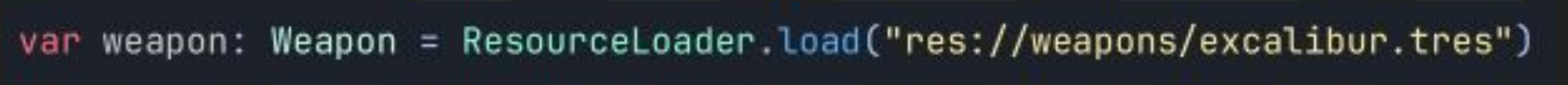
```
└ weapons
    ├ excalibur.tres
    ├ gunblade.tres
    ├ lightsaber.tres
    ├ mjolnir.tres
    └ weapon.gd
```

Resources can have functions

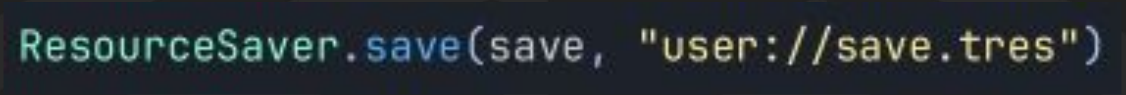


```
class_name Weapon
extends Resource

@export var name: String
@export var damage: float
@export var range: float
@export var area_radius: float
@export var texture: Texture2D
@export var sound_effect: AudioStream
@export var particles: ParticleProcessMaterial


func upgrade() -> void:
	pass  # Note: Upgrade code can go here

```

Resources can be loaded/saved

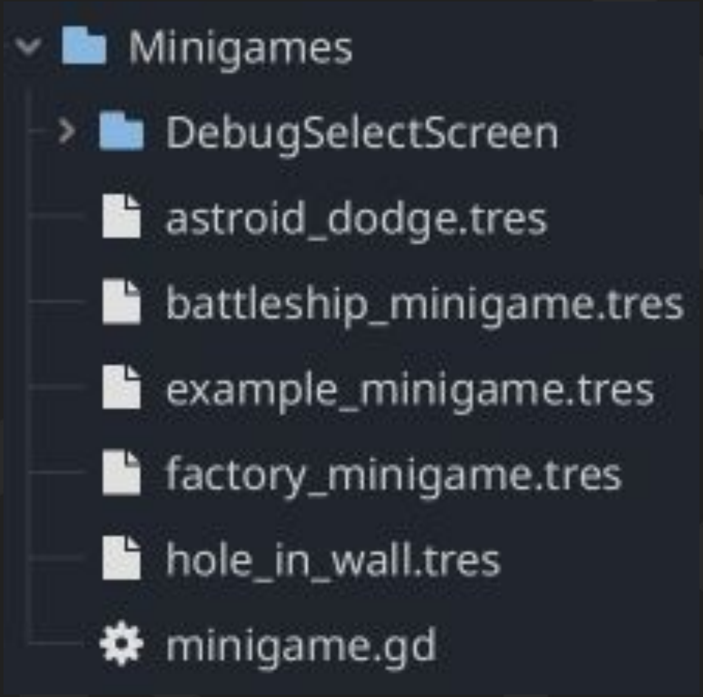


```
var weapon: Weapon = ResourceLoader.load("res://weapons/excalibur.tres")
```

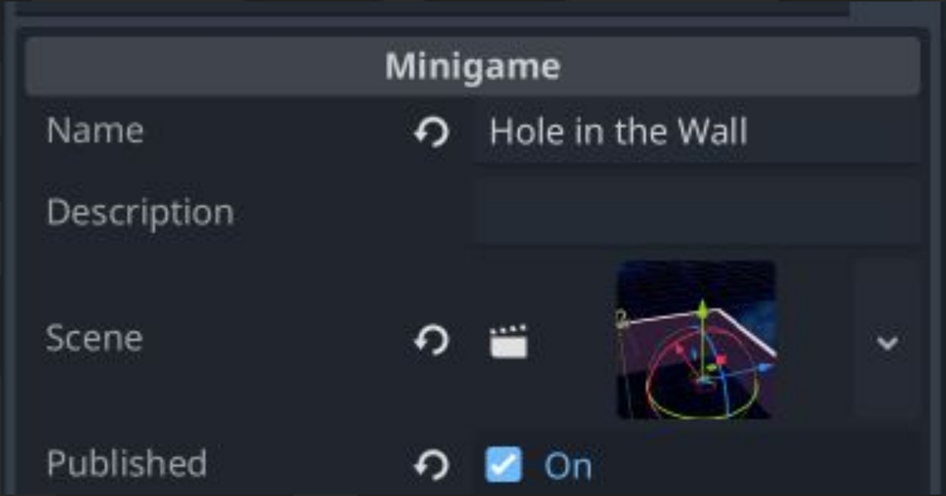


```
ResourceSaver.save(save, "user://save.tres")
```

Another example custom resource



- Minigames
  - DebugSelectScreen
  - astroid_dodge.tres
  - battleship_minigame.tres
  - example_minigame.tres
  - factory_minigame.tres
  - hole_in_wall.tres
  - minigame.gd



Minigame

Name	Hole in the Wall
Description	
Scene	
Published	✓ On

Autoloads

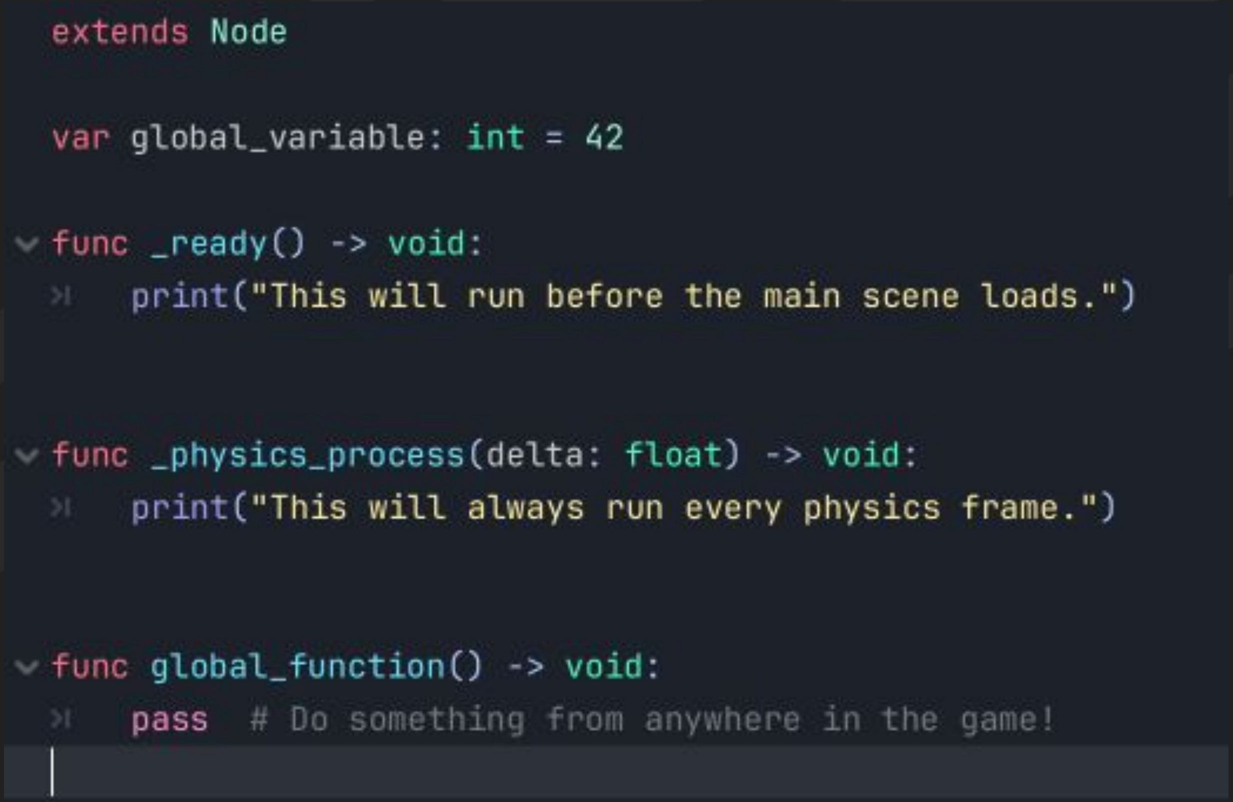
What is an autoload?

- Sometimes you need to store information used by more than one scene.
  - E.g., a player's score, inventory

- An autoload is a node (or scene) that is always loaded in the game.
- Essentially the "singleton" programming pattern.

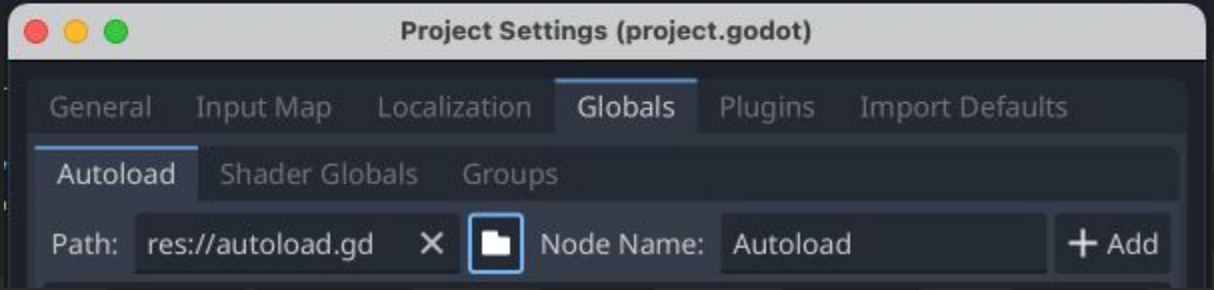
- There are some other ways to do this:
  - Use a "main" scene that loads and unloads other scenes as its children.
    - Now child scenes may depend on the main scene.
  - Save and load data to files on disc.
    - Complicates the code and may slow down the game.

Creating an autoload



```gdscript
extends Node

var global_variable: int = 42

func _ready() -> void:
	print("This will run before the main scene loads.")


func _physics_process(delta: float) -> void:
	print("This will always run every physics frame.")


func global_function() -> void:
	pass  # Do something from anywhere in the game!
```

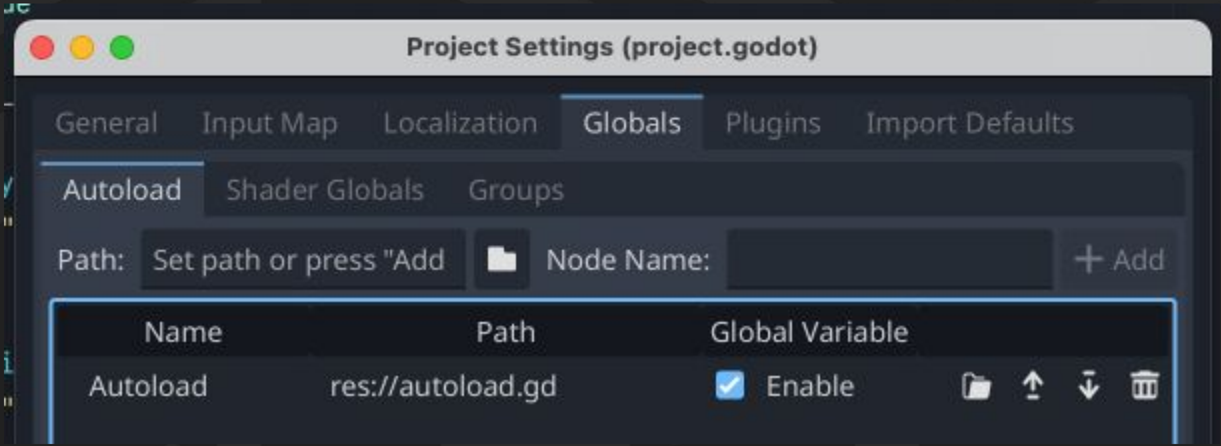
Creating an autoload


Project Settings (project.godot)

General | Input Map | Localization | Globals | Plugins | Import Defaults

Autoload | Shader Globals | Groups

Path: res://autoload.gd Node Name: Autoload + Add


Project Settings (project.godot)

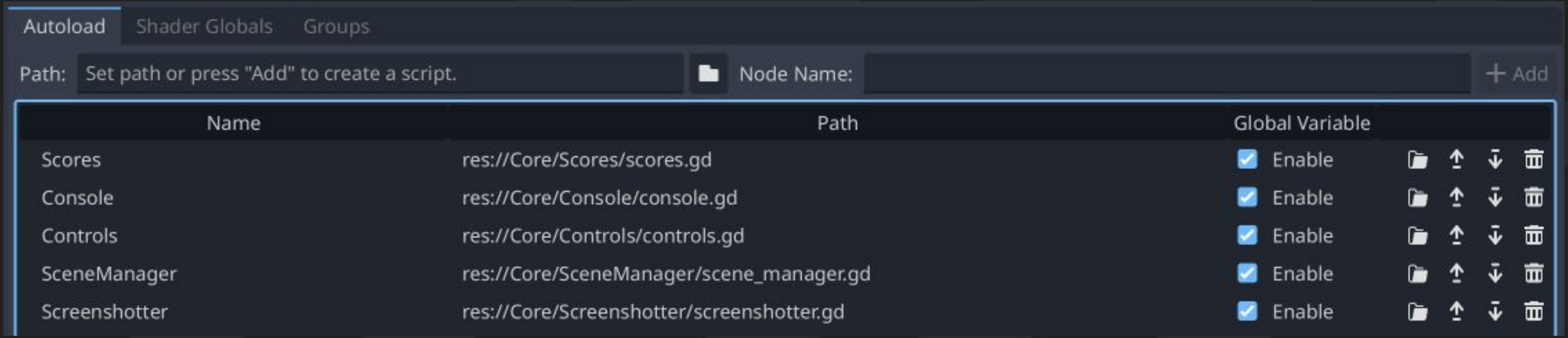
General | Input Map | Localization | Globals | Plugins | Import Defaults

Autoload | Shader Globals | Groups

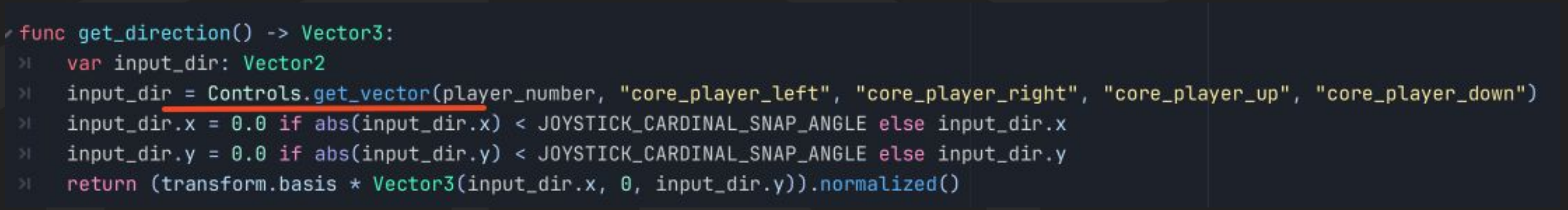
Path: Set path or press "Add Node Name: + Add

Name	Path	Global Variable
Autoload	res://autoload.gd	✓ Enable

Example autoloads (Orbital Odyssey)



Name	Path	Global Variable
Scores	res://Core/Scores/scores.gd	✓ Enable
Console	res://Core/Console/console.gd	✓ Enable
Controls	res://Core/Controls/controls.gd	✓ Enable
SceneManager	res://Core/SceneManager/scene_manager.gd	✓ Enable
Screenshotter	res://Core/Screenshotter/screenshotter.gd	✓ Enable



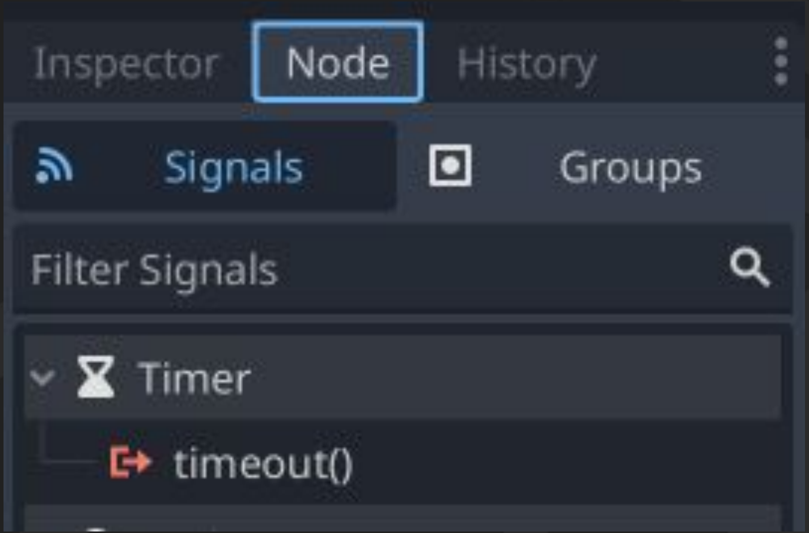
```
func get_direction() -> Vector3:
    var input_dir: Vector2
    input_dir = Controls.get_vector(player_number, "core_player_left", "core_player_right", "core_player_up", "core_player_down")
    input_dir.x = 0.0 if abs(input_dir.x) < JOYSTICK_CARDINAL_SNAP_ANGLE else input_dir.x
    input_dir.y = 0.0 if abs(input_dir.y) < JOYSTICK_CARDINAL_SNAP_ANGLE else input_dir.y
    return (transform.basis * Vector3(input_dir.x, 0, input_dir.y)).normalized()
```

Signals

What is a signal?

- A signal is a message that a node can emit when something happens.
  - E.g., a button being pressed
- Nodes can connect to that signal to run code when that something happens.

Can be connected through the editor

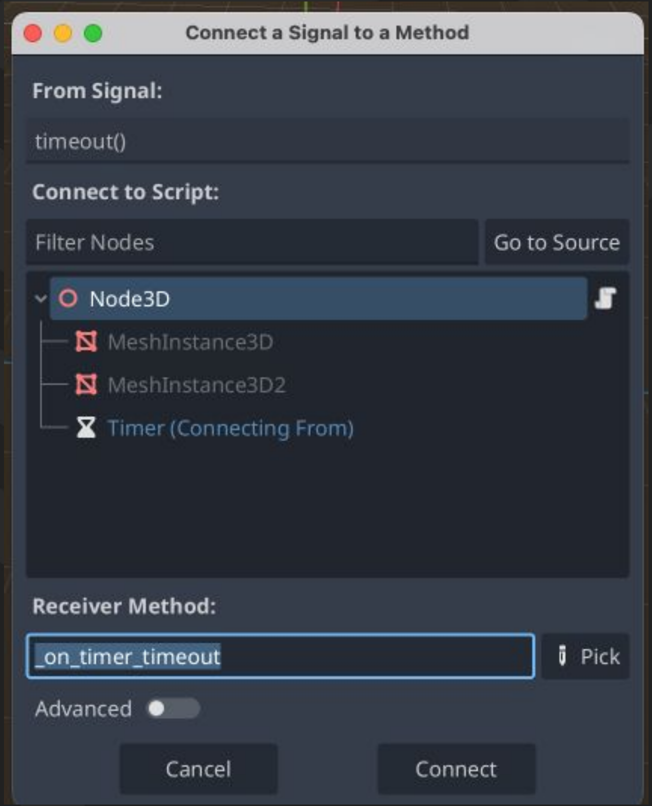

Inspector Node History

Signals Groups

Filter Signals

Timer

timeout()


Connect a Signal to a Method

From Signal:

timeout()

Connect to Script:

Filter Nodes Go to Source

Node3D

MeshInstance3D

MeshInstance3D2

Timer (Connecting From)

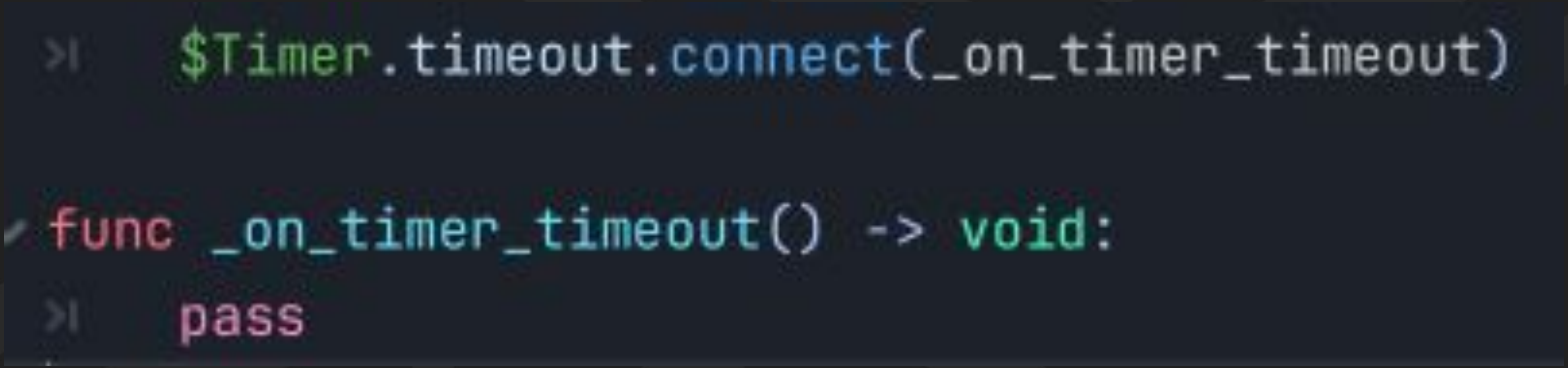
Receiver Method:

_on_timer_timeout Pick

Advanced

Cancel Connect

Can be connected through a script



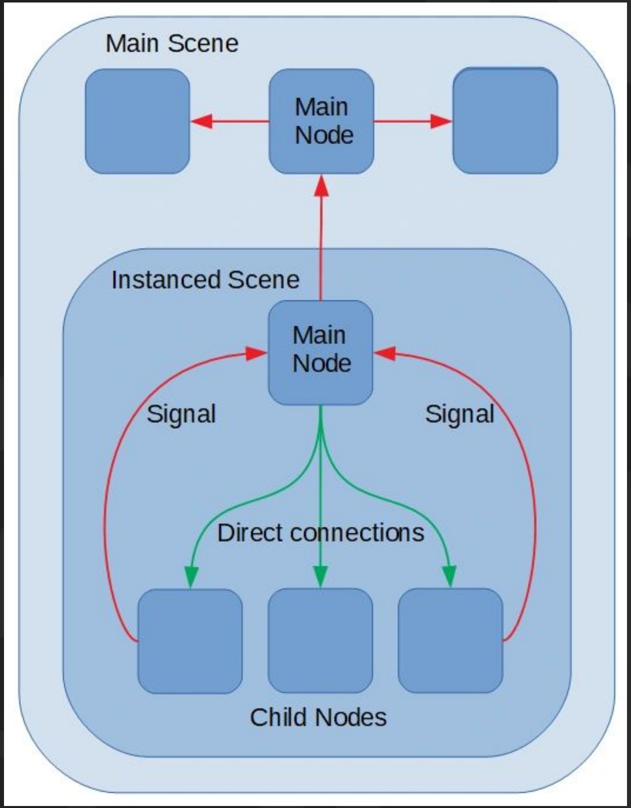
```
	$Timer.timeout.connect(_on_timer_timeout)

func _on_timer_timeout() -> void:
	pass
```

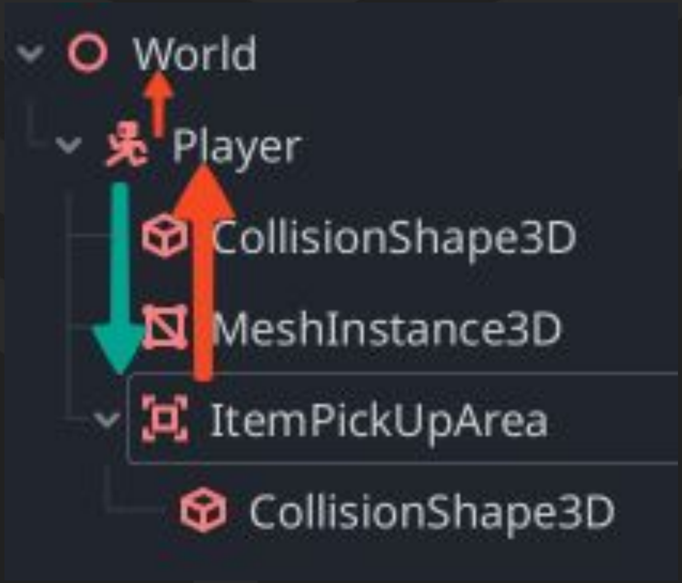
Can create custom signals

```
extends Node

signal my_signal
```

```
extends Node

signal my_signal(int, String, Texture2D, Weapon)
```

“Call down, signal up”



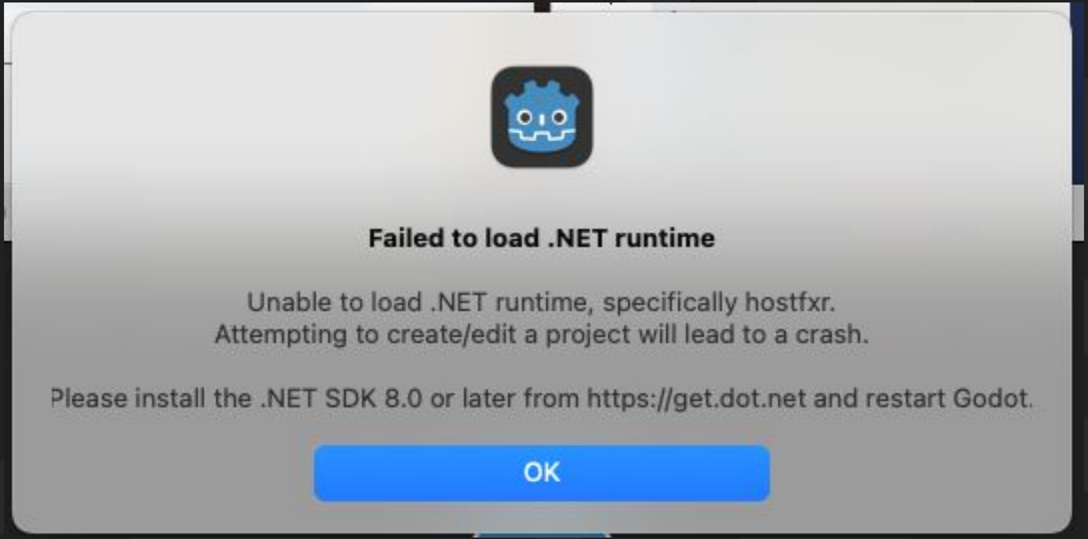


Plugins

Importing Assets

Using C#

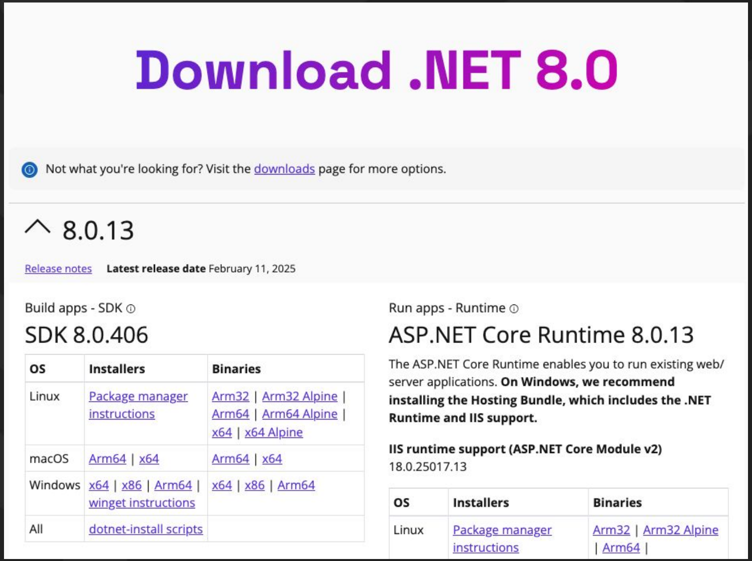
To use C#, you need to download .NET 8.0



Failed to load .NET runtime

Unable to load .NET runtime, specifically hostfxr.
Attempting to create/edit a project will lead to a crash.

Please install the .NET SDK 8.0 or later from https://get.dot.net and restart Godot.



Download .NET 8.0

Not what you're looking for? Visit the downloads page for more options.

8.0.13

Release notes **Latest release date** February 11, 2025

Build apps - SDK

SDK 8.0.406

OS	Installers	Binaries
Linux	Package manager instructions	Arm32 | Arm32 Alpine | Arm64 | Arm64 Alpine | x64 | x64 Alpine
macOS	Arm64 | x64	Arm64 | x64
Windows	x64 | x86 | Arm64 | winget instructions	x64 | x86 | Arm64
All	dotnet-install scripts	

Run apps - Runtime

ASP.NET Core Runtime 8.0.13

The ASP.NET Core Runtime enables you to run existing web/server applications. **On Windows, we recommend installing the Hosting Bundle, which includes the .NET Runtime and IIS support.**

IIS runtime support (ASP.NET Core Module v2)
18.0.25017.13

OS	Installers	Binaries
Linux	Package manager instructions	Arm32 | Arm32 Alpine | Arm64 |

Godot v4.4

https://godotengine.org/releases/4.4/

Sign in here!

fsu.devlup.org/signin

DevLUp FSU

GBM #6

Godot Tidbits

March 6th, 2025