Godot!

RGDC Discord

Jam page

What’s godot?

- A FOSS game engine (ofc why else would I be talking about it here)
  - MIT license
- 2D and 3D
- C#, GDScript, and many other language bindings through GDExtension
  - Rust, Swift, C++
- Available for Linux, Windows, MacOS, Android, and on the web.
  - Supposedly works on BSD too
- Can export to Linux, Windows, MacOS, iOS, Android, and web
- Yes you can also compile from source https://docs.godotengine.org/en/stable/contributing/development/compiling/compiling_for_linuxbsd.html

I’m not a game developer, this isn’t RGDC, Google slides isn’t FOSS why are you using it?

- Godot can be used for developing cross-platform software and tools!
  - https://itch.io/c/651672/tools-made-with-godot-engine
  - https://godotengine.org/showcase/rpg-in-a-box/
  - https://godotengine.org/showcase/dungeondraft/
  - https://godotengine.org/showcase/material-maker/
  - https://godotengine.org/showcase/pixelorama/
- Godot itself is kind of a godot game
  - https://docs.godotengine.org/en/stable/getting_started/introduction/godot_design_philosophy.html#the-godot-editor-is-a-godot-game

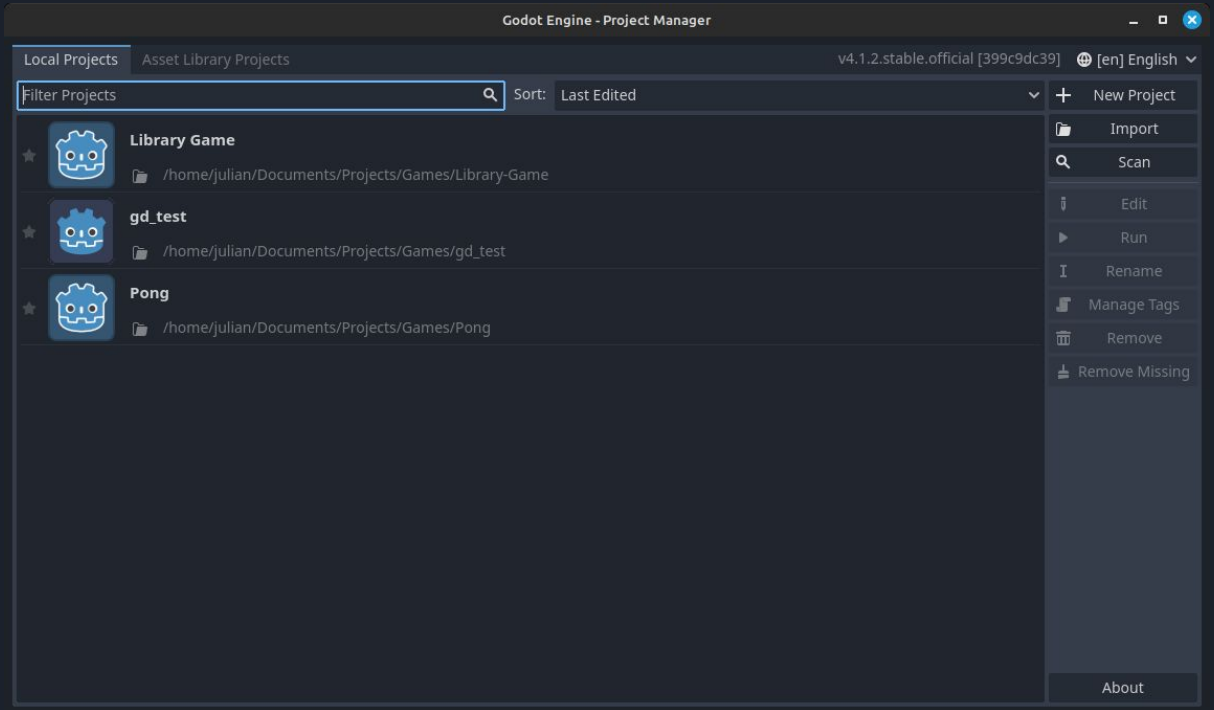
Getting started

1. Go to https://godotengine.org/
2. Click “download latest” (download the .NET version for C# support)
3. Run the executable

(You can also get it from most package managers, itch, steam, and the EGS… but like, why?)

The Project Manager

It manages projects



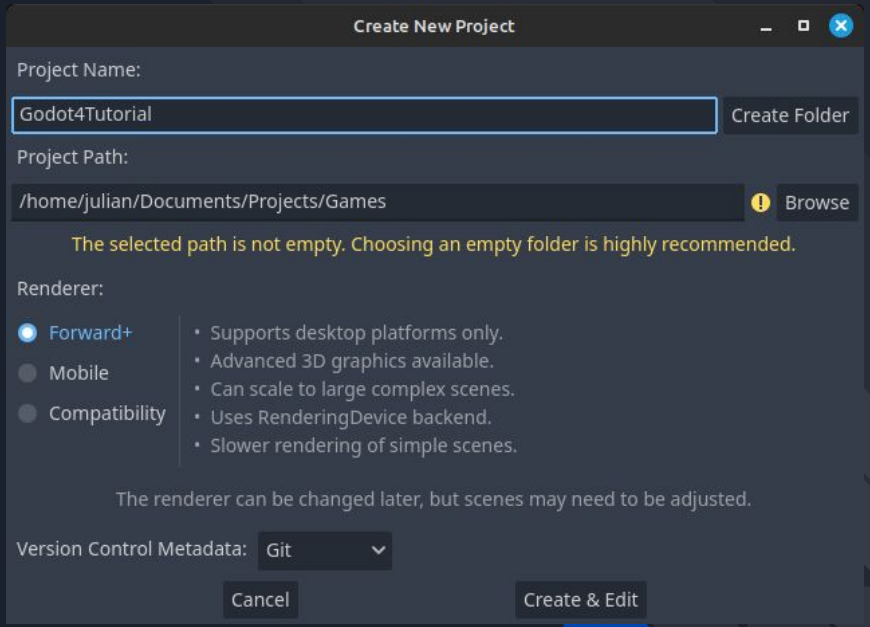
Click “New Project”

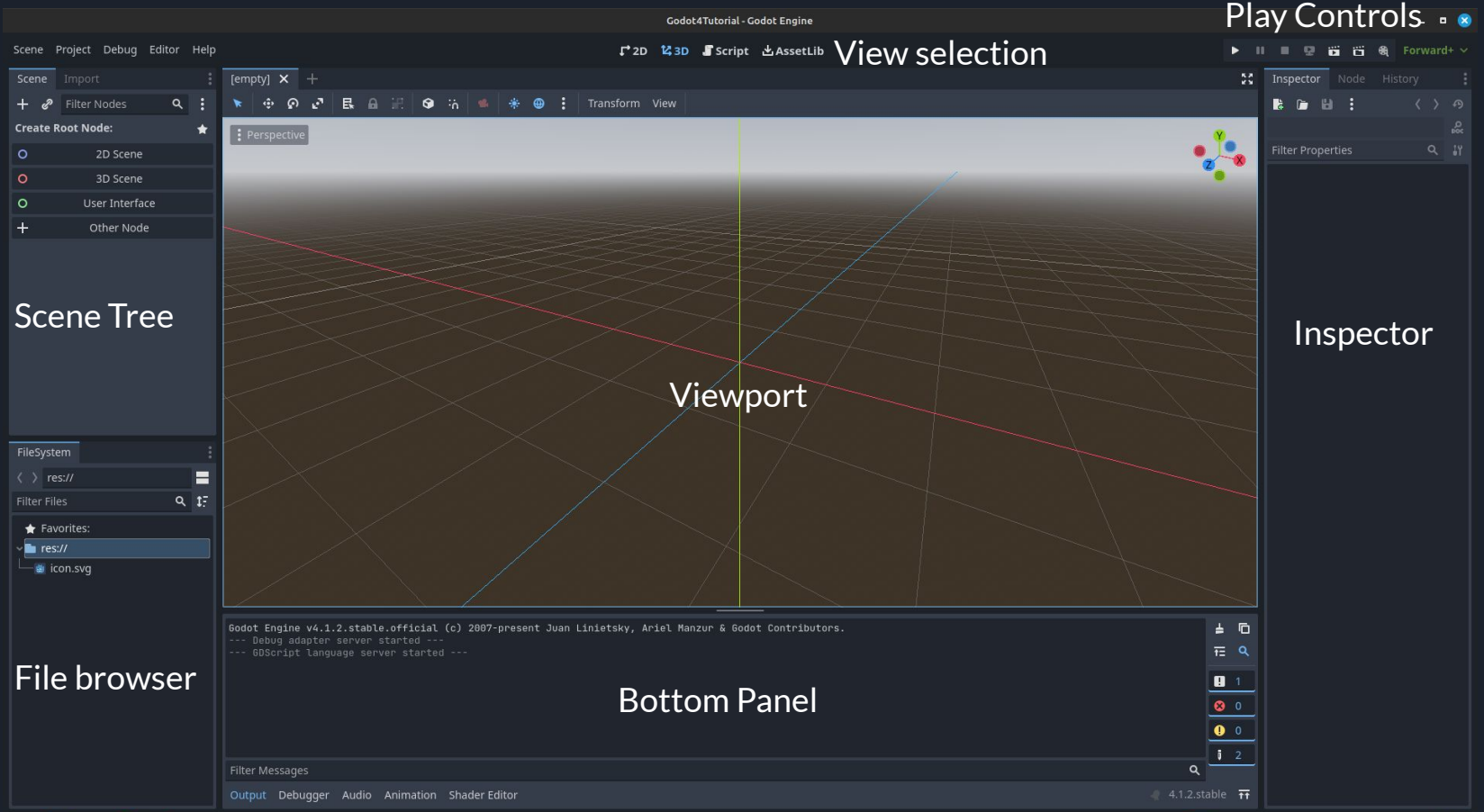
Select a path

Choose a name

Click “Create folder”

Click “Create & Edit”




Godot4Tutorial - Godot Engine
Play Controls
Scene Project Debug Editor Help
2D 3D Script AssetLib
View selection
Forward+
Scene Import
Filter Nodes
Create Root Node:
2D Scene
3D Scene
User Interface
Other Node
Scene Tree
[empty]
Transform View
Perspective
Viewport
Inspector Node History
Filter Properties
Inspector
FileSystem
res://
Filter Files
Favorites:
res://
icon.svg
File browser
Godot Engine v4.1.2.stable.official (c) 2007-present Juan Linietsky, Ariel Manzur & Godot Contributors.
--- Debug adapter server started ---
--- GDScript language server started ---
Bottom Panel
Filter Messages
Output Debugger Audio Animation Shader Editor
4.1.2.stable

3 main concepts

Nodes

Resources

Signals

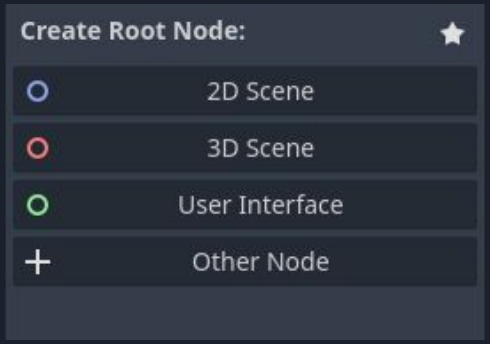
Nodes

- Analogous to GameObjects, Scenes, *and* components in Unity
- Organized in a tree structure
- All scenes have a root node
- To add functionality, create child nodes
- Getting nodes by path with get_node() or $pathname
- Nodes inherit from other nodes
- It's possible to define your own node types!

Class: Sprite2D

Inherits: Node2D < CanvasItem < Node < Object

Creating a scene



- Click “2D scene” to create a Node2D at the root.
- Control+S to save to a file

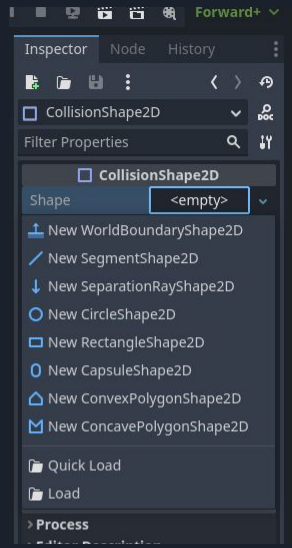
Creating a character controller

- Godot has a CharacterBody2D node, which is a 2d physics body meant for character movement
- The CharacterBody2D node requires a CollisionShape2D node as a child.

- You can use the options along the top to move nodes with a transform (or use the inspector)

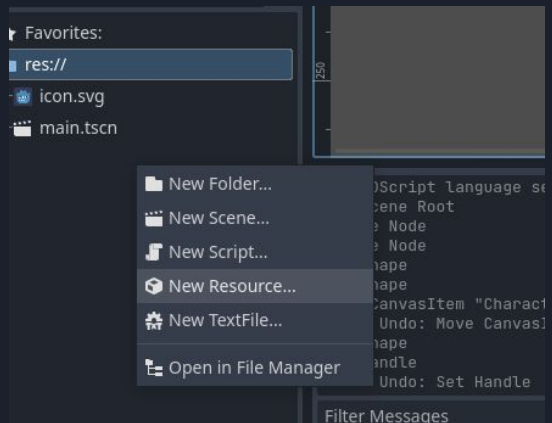
Resources

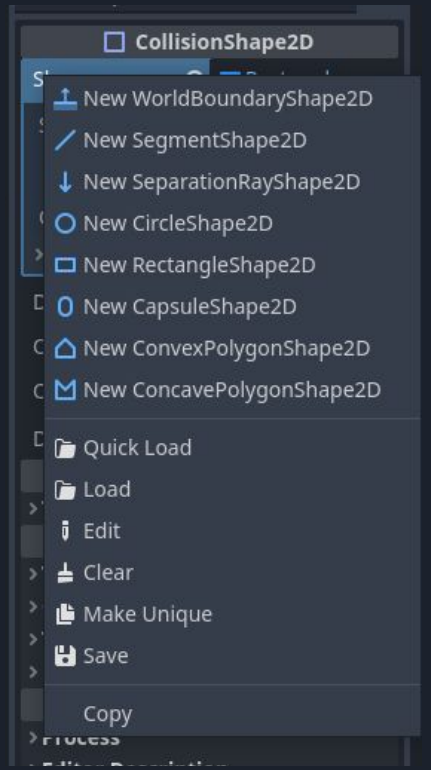
- Represent any data
- Many types build-in:
  - Animation, Colliders, Meshes, Curves, Fonts, Textures, Materials, Key Shortcuts
- You can even create your own to store custom data!
  - Similar to scriptable objects in Unity (But Godot can serialize dictionaries!)
- To function, our CollisionShape2D needs to know what collision shape to use
- Afterwards, we can expand to change its properties



More about resources

- The collision shape we made is saved in the scene file
- But we can save resources to files to reuse them!
- You can also copy and paste resources
  - Be careful - if you paste a resource somewhere else and change it, it will also change the original!
  - Use the “Make Unique” option in the right click menu to avoid this.
- Can also create resources in the file manager

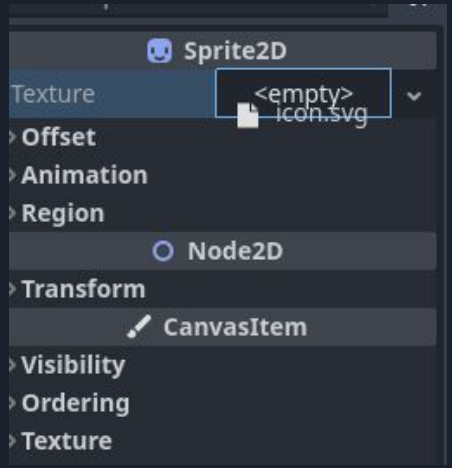




Giving the character some visuals

- Let's add a Sprite2D node to our character
- We can chose the texture resource manually…

- …or just drag an image in



* make sure the collider size and sprite match!

Making a script

- Right click in file manager -> new script
- Choose a name
- Open the scripts view using the button at the top

GDScript

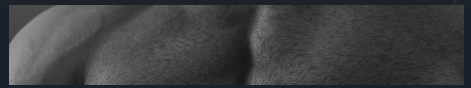
- Similar to Python and Lua
- Scripts extend a certain node type, which determines which nodes they can be attached to
- Lots of nice features to make game development really easy
- In Unity, _ready is analogous to Start, _process is analogous to Update

```
extends Node


# Called when the node enters the scene tree for the first time.
func _ready():
	pass # Replace with function body.


# Called every frame. 'delta' is the elapsed time since the previous frame.
func _process(delta):
	pass
```

Writing a top-down character controller

- Input maps are defined in Project->Project Settings->Input Map (We'll use the built-in UI actions for simplicity though)
- "Ugh we have to make if statements for each of the four inputs to see if they're pressed and sum together into a vector and then normalize it so you can't go faster when going diagonal"

```
func _process(delta):
    var input_direction = Input.get_vector("ui_left", "ui_right", "ui_up", "ui_down")
```



Moving the character

- First we need to inherit from CharacterBody2D to get access to velocity
- Set the velocity

```
func _process(delta):
    var input_direction = Input.get_vector("ui_left",
    velocity = input_direction * 400
    move_and_slide()
```

- Call “move_and_slide”
  - The CharacterBody2D doesn’t move based on physics - this tells it to move, and slide against any colliders
- Now, let’s add our script to the CharacterBody2D node (click and drag)
- Run!

Making it a 2d platformer

- First let's rename nodes to be more clear
  - Let's name our CharacterBody2D node "Player"
  - While we're at it, save this as a scene so we can reuse it

Adding gravity

- We can define const values
- Or make it an export so we can change it in the editor
- Use “var” to define a variable
- Use type hints to speed up code (required for exports)

```
@export var GRAVITY : int = 200
@export var speed : int = 400
```

Our 2d character movement script

```
func _process(delta):
    var input_direction = Vector2(Input.get_axis("ui_left", "ui_right") * speed, GRAVITY)
    velocity = input_direction * delta
    move_and_slide()
```

Add some platforms

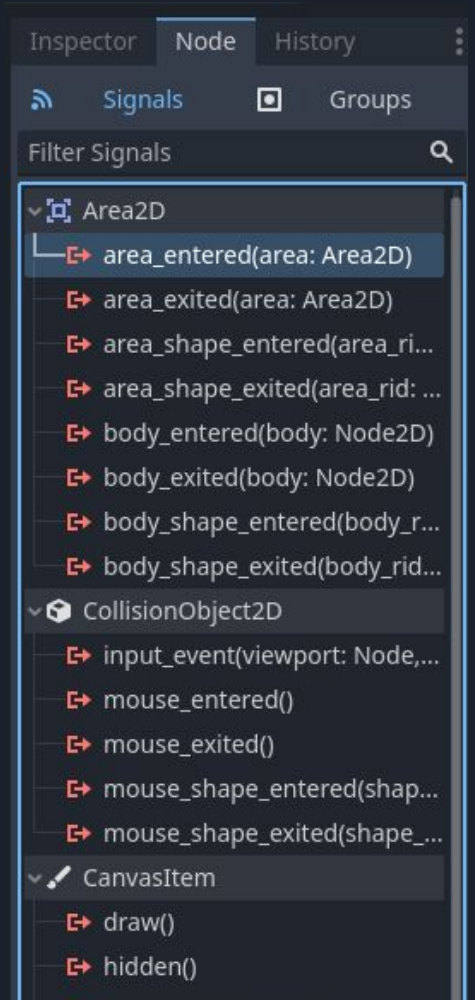
- StaticBody2D is for physics objects that don't move

Signals

- Used to broadcast events or send data to other nodes
- Built-in nodes usually have many signals
- As always, you can define your own

Using Signals

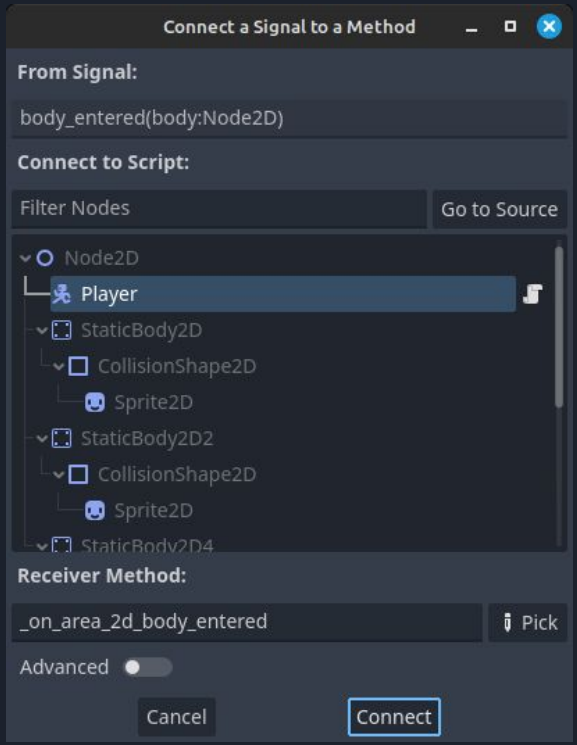
- Let's make a gravity reverse trigger
- Create an Area2D - it can detect if objects enter an area
- In the inspector, select the "Node" tab
- There are a lot of signals



Detect entering an area

- We want a signal sent when the area is entered
- Select the node with the script we want to call a function of
  - In our case, we only have the player script
- It will create a function if it doesn't exist, but you can use an existing function as well
- Basic collider check:

```
func _on_area_2d_body_entered(body : Node2D):
    if body == self:
        GRAVITY *= -1
```



Detect leaving an area

- We can use the body_exited signal

```
29 func _on_area_2d_body_entered(body : Node2D):
30     if body == self:
31         GRAVITY *= -1
32
33
34 func _on_area_2d_body_exited(body):
35     if body == self:
36         GRAVITY *= -1
```

* The green arrow means its connected to one or more signals - click it to see all of them!

Cool Stuff about Godot

- Press F1 to start searching documentation for anything in Godot
  - Or control+click stuff in your code
- Make your own docs with double comments!

```
extends CharacterBody2D
## This class does things

## The gravity
@export var GRAVITY : int

## The speed!
@export var speed : int
```

Class: "player_movement.gd"

Inherits: CharacterBody2D < PhysicsBody2D < CollisionObject2D < Node2D < CanvasItem < Node < Object

This class does things

Properties

int GRAVITY
int speed

Property Descriptions

- int GRAVITY

  The gravity

- int speed

  The speed!

Cool Stuff (cont.)

- Godot's UI tools are very extensive
  - Godot's own UI is made with Godot's UI elements!
  - When creating editor plugins, you can use Godot's UI editor to create the UI for your plugin!
- Click+drag any properties from the inspector into your code to get their names.

Cool Stuff (cont.)

- Updates!
  - Godot is constantly getting huge updates right now, thanks to increasing interest and some sizable donations.