Game programming with Godot

foss-gbg 2019-10 // Johan Thelin

What we will look at today

- Intro to the editor
- 2D and 3D games
- Intro to VR using Godot
- Intro to deployment using Godot

Intro

- I encourage you to code along!
- All examples can be found at https://github.com/e8johan/godot-tutorial .

Assets	the raw assets
01_flippable	2D intro example
02_platformer	2D platformer
02b_platformer_animated	2D platformer with animated hero
03_on_a_roll	3D example
04_on_a_roll_vr	3D VR example

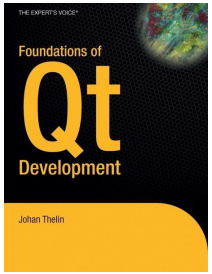
Bio

Johan Thelin - co-founder of Kuro Studio

Autoliv → XDIN → Bitsim → Trolltech → Pelagicore → Kuro

I’ve done lots and lots of embedded devices with Qt, Linux, etc

Absolutely zero game programming experience :-)



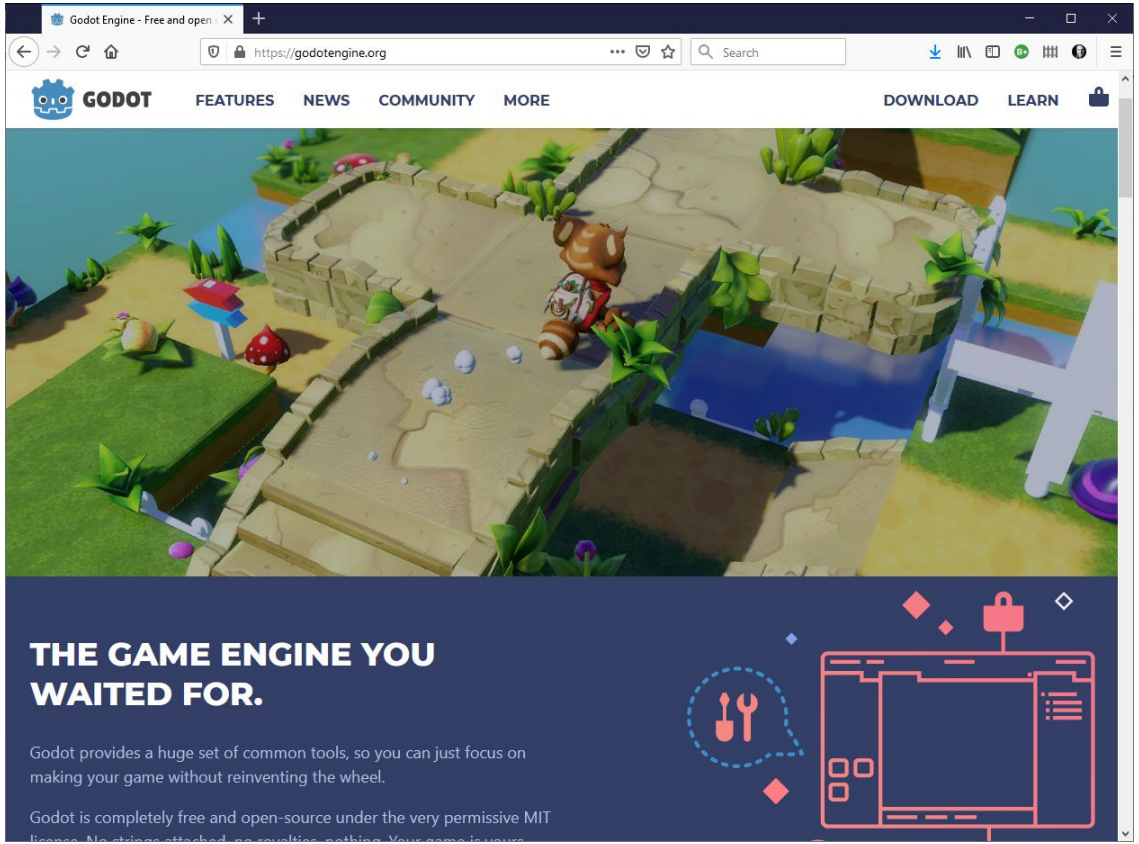
What is Godot

Modern game engine

Visual editor

Open source

www.godotengine.org



Getting Godot

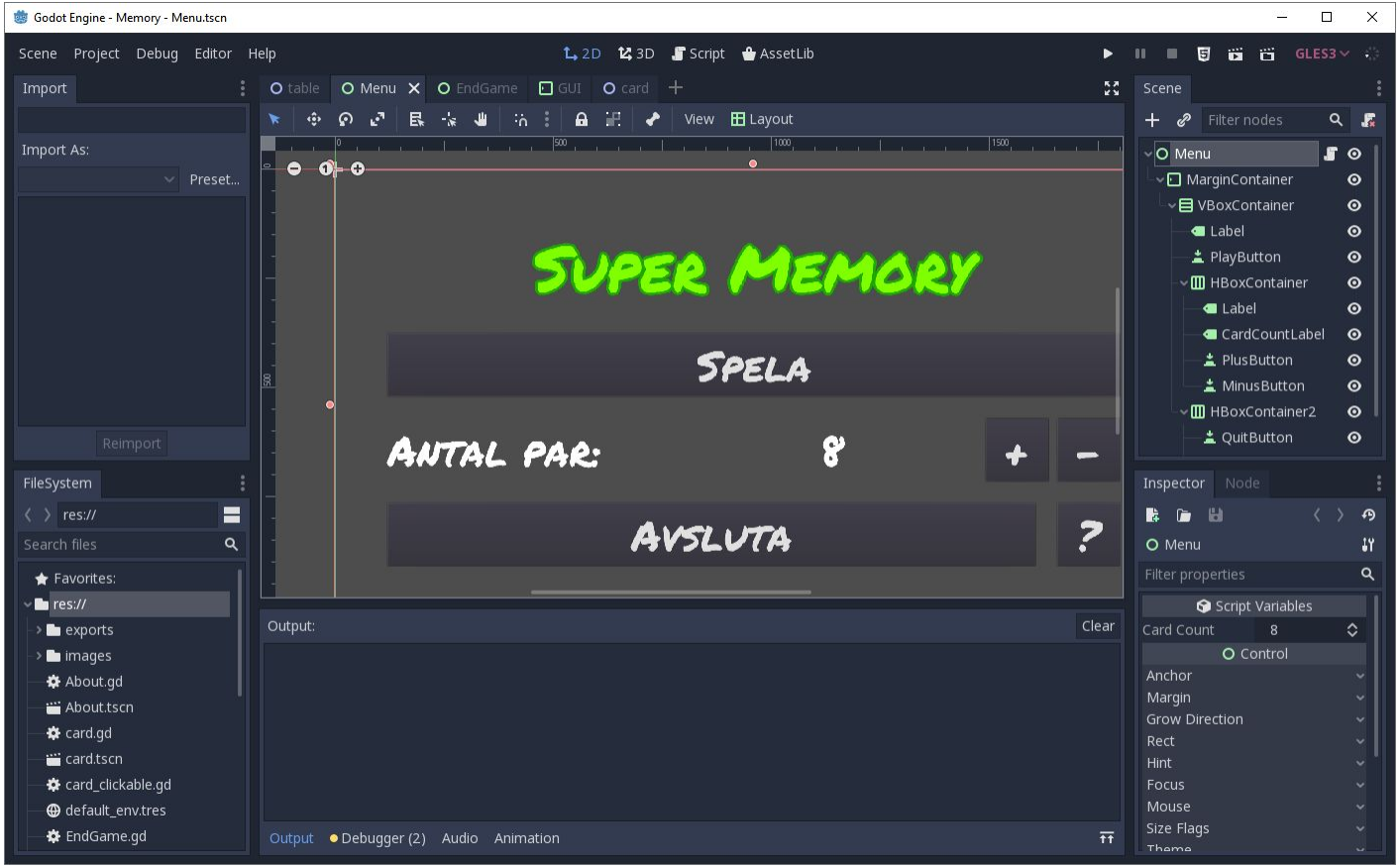
- A ~25MB download from https://godotengine.org/download
- Available for Linux, Windows, MacOS and Server

- The download contains the executable - just download and run
- … or use your distro's package manager

- Today, we will focus on the standard version - no C# or other external deps
- I'm using version 3.1

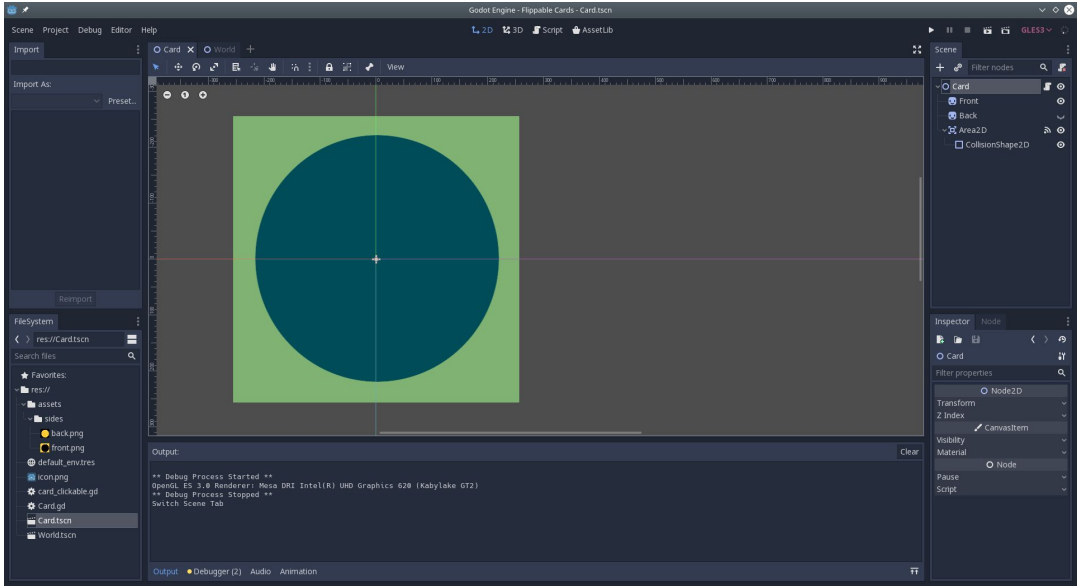
The Editor

- Main view
  - 2D
  - 3D
  - Script
- File system
  - res://
- Scene
- Inspector



Let’s create!

- You will learn about
  - Nodes
  - Scenes
  - Scripts



- Create nodes
  - Node2D
    - Sprite x2
    - Area2D
      - CollisionShape2D

The flipper script

```
extends Node2D

onready var front = $Front
onready var back = $Back

func _on_Area2D_input_event(viewport, event, shape_idx):
    if event is InputEventMouseButton:
        if event.is_pressed() and event.button_index == BUTTON_LEFT:
            front.visible = !front.visible
            back.visible = !front.visible
```

GDScript

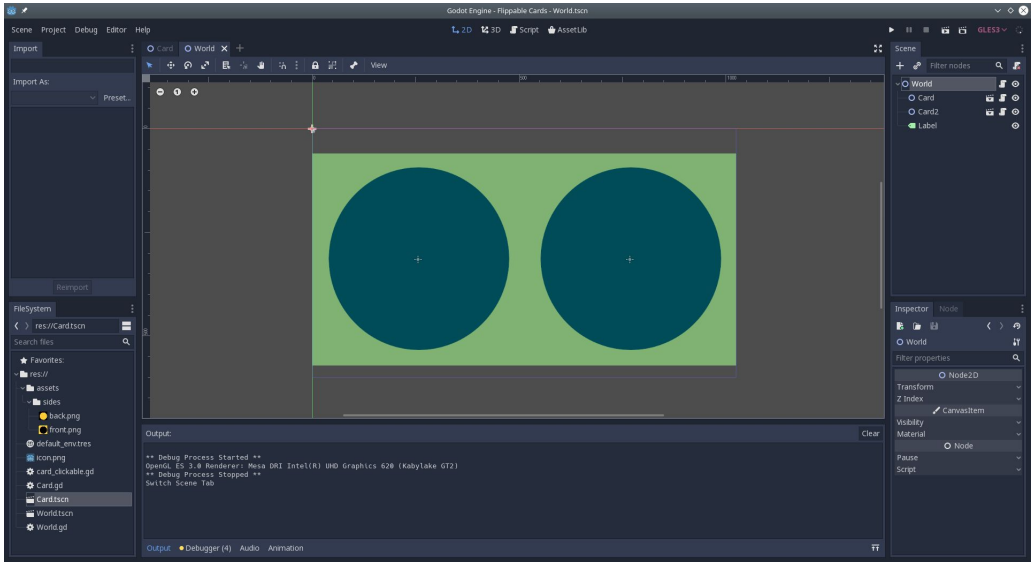
- Very pythonesque - but not Python
- Can do typing - will improve performance
  - `var x : int = 10`
  - `var y := calculate_something()`
  - `func double_it(value : int) -> int:`
  - Enums are problematic

- Resources I go to for help
  - https://docs.godotengine.org/en/3.1/getting_started/scripting/gdscript/gdscript_basics.html
  - https://www.gdquest.com/open-source/guidelines/godot-gdscript/
- There is more - my code is far from stylistically correct or efficient

Creating a world

- Create a world with two cards, and a UI

- Update UI based on state

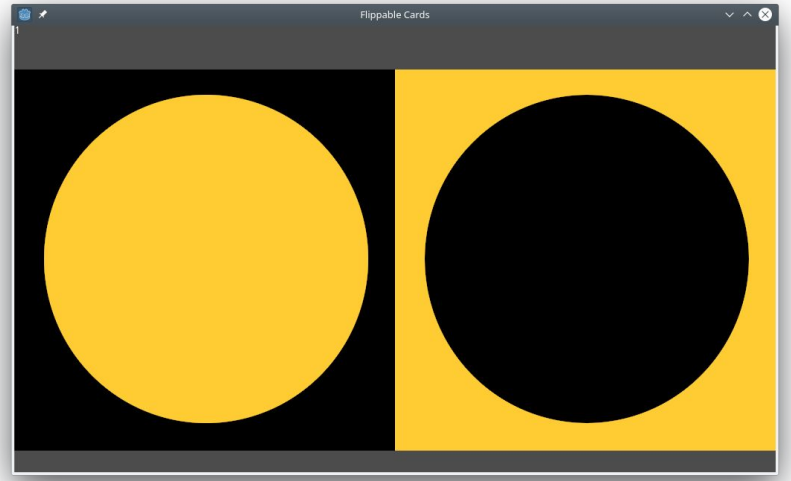


Signal and setter for the Card

```
extends Node2D

signal flipped_changed
var flipped : bool = false setget set_flipped

onready var front = $Front
onready var back = $Back

func _on_Area2D_input_event(viewport, event, shape_idx):
    if event is InputEventMouseButton:
        if event.is_pressed() and event.button_index == BUTTON_LEFT:
            set_flipped(front.visible)

func set_flipped(f : bool) -> void:
    front.visible = !f
    back.visible = f
    flipped = f
    emit_signal("flipped_changed")
```

The World script

```
extends Node2D

func _ready():
    $Label.text = str(no_of_flipped_cards())
    $Card.connect("flipped_changed", self, "_on_flipped_changed")
    $Card2.connect("flipped_changed", self, "_on_flipped_changed")

func _on_flipped_changed():
    $Label.text = str(no_of_flipped_cards())

func no_of_flipped_cards() -> int:
    var res : int = 0
    if $Card.flipped:
            res += 1
    if $Card2.flipped:
            res += 1
    return res
```



Finding children - the smarter way

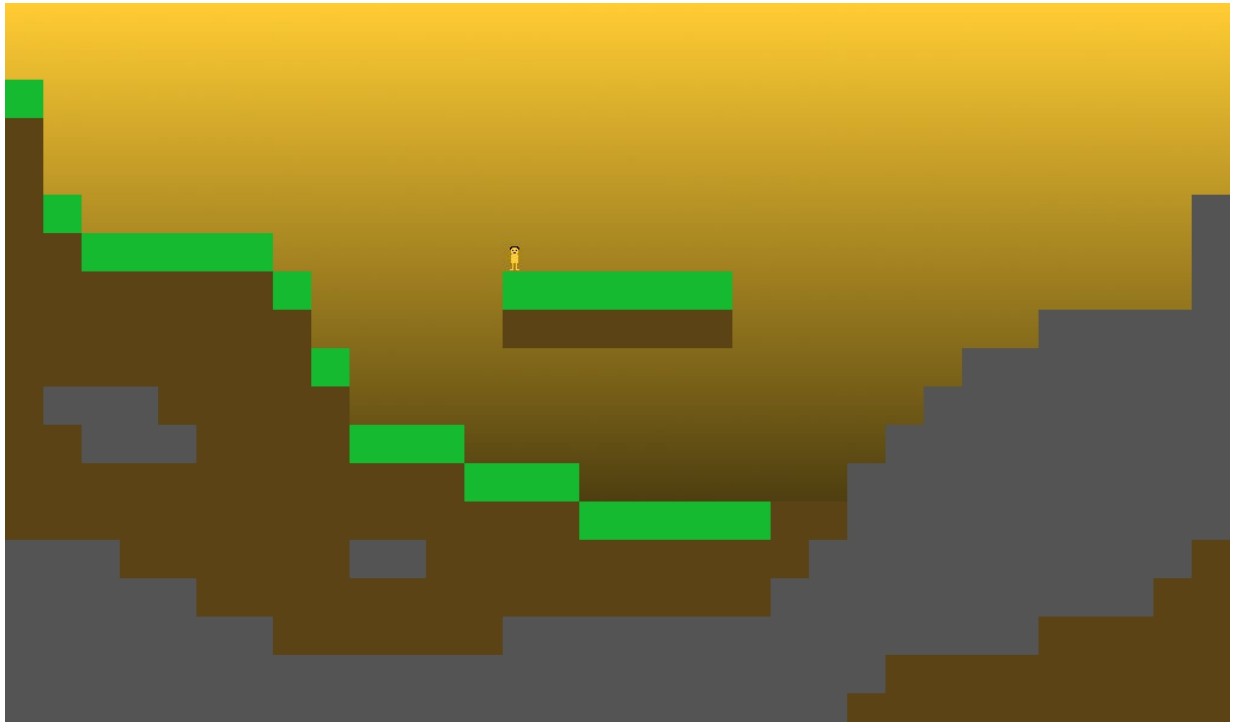
```
func _find_all_cards() -> Array:
    var res : Array = []
    var children : = self.get_children()
    for child in children:
        if child.is_in_group("cards"):
                res.append(child)
    return res
```

From https://github.com/e8johan/supermemory/blob/master/table.gd#L210

Platformer time!

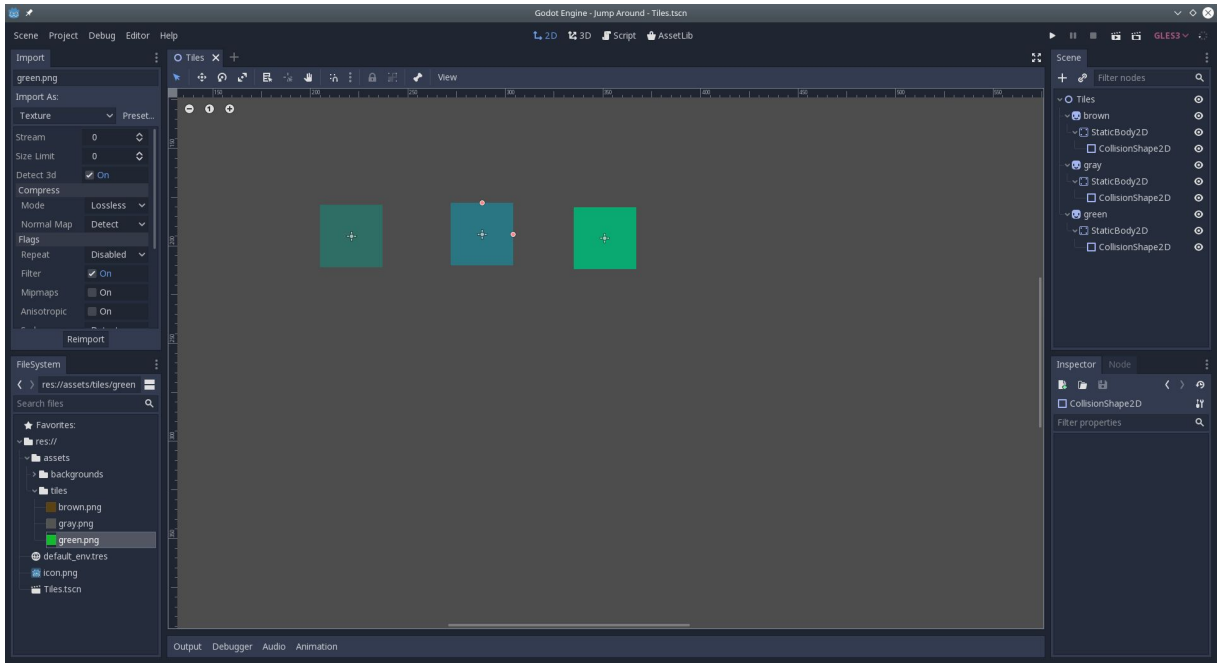
- The game plan:
  - Create tile set
  - Create tile map
  - Create character
  - Jump around

- You will learn about
  - Tiled 2D worlds
  - 2D movement
  - Animations
  - Input events



The tile set

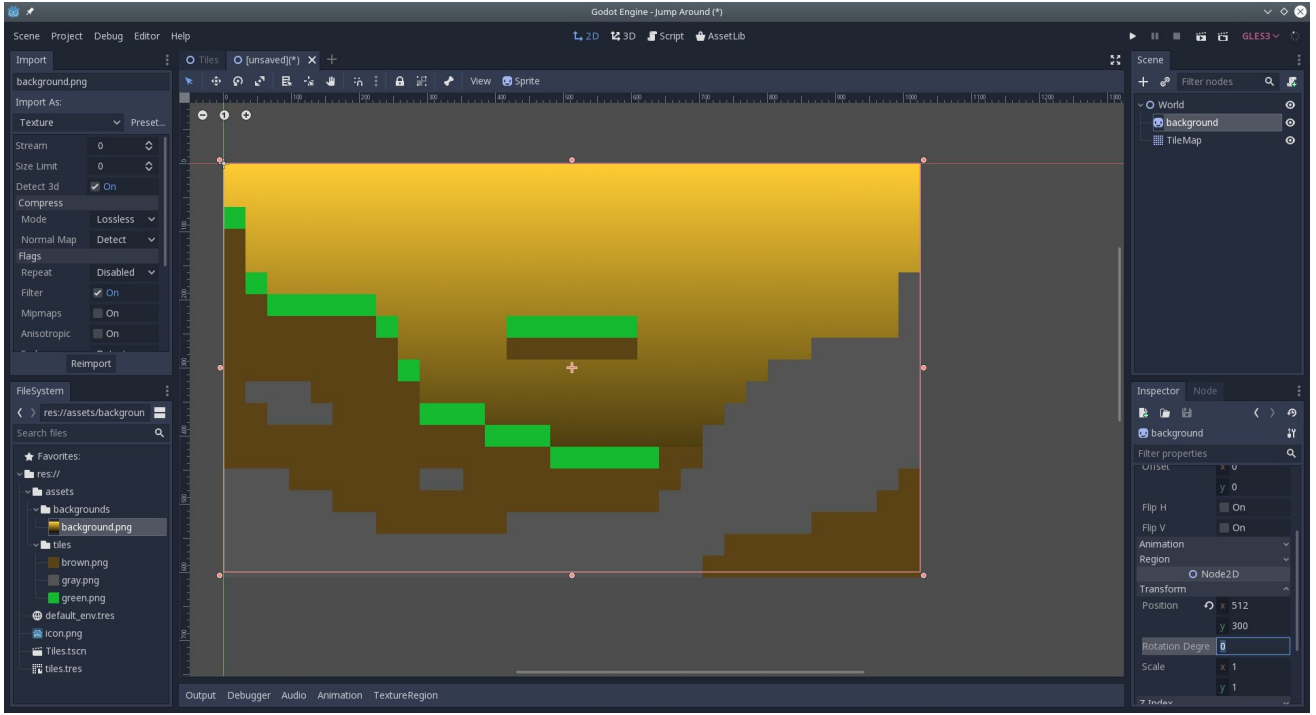
- 2D Scene - “Tiles”
  - Sprite
  - StaticBody2D
  - CollisionShape2D
  - RectangleShape2D
  - Extent



- Scene → Convert to… → TileSet...

The tile map

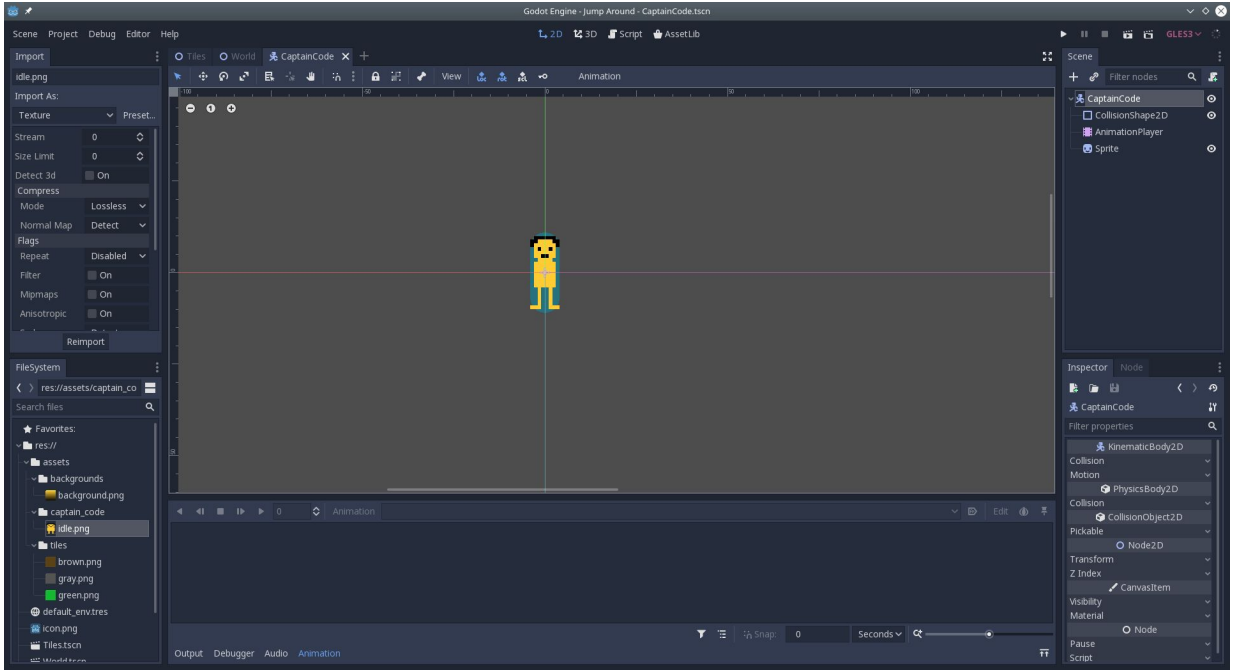
- 2D Scene - World
- TileMap
  - Cell size
- Sprite - for bg



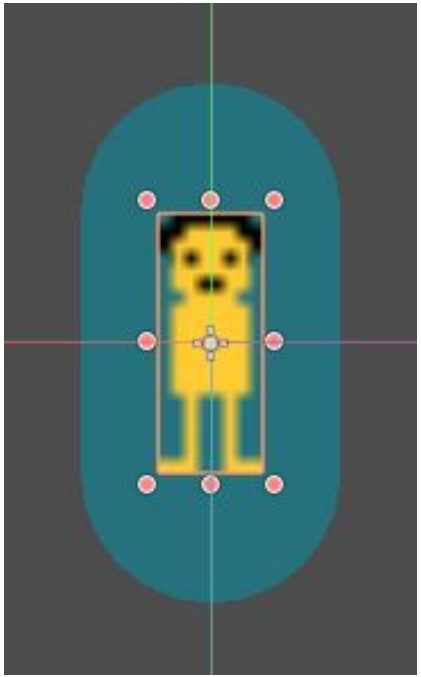
Enter: the hero!

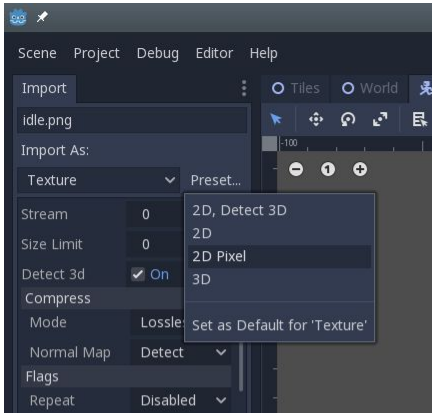
- KinematicBody2D
  - CollisionShape2D
    - CapsuleShape
  - Sprite

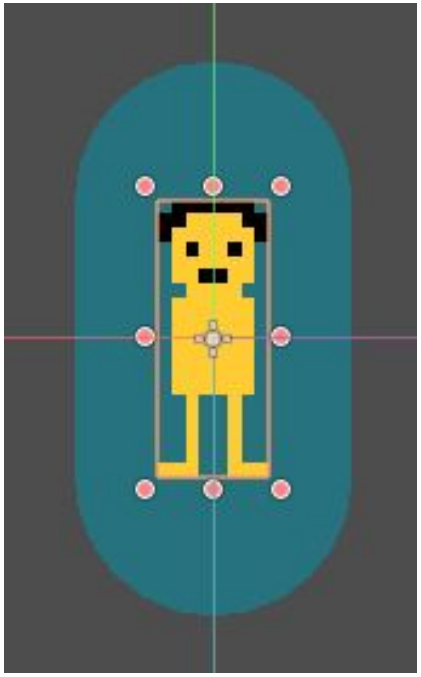
- CaptainCode!



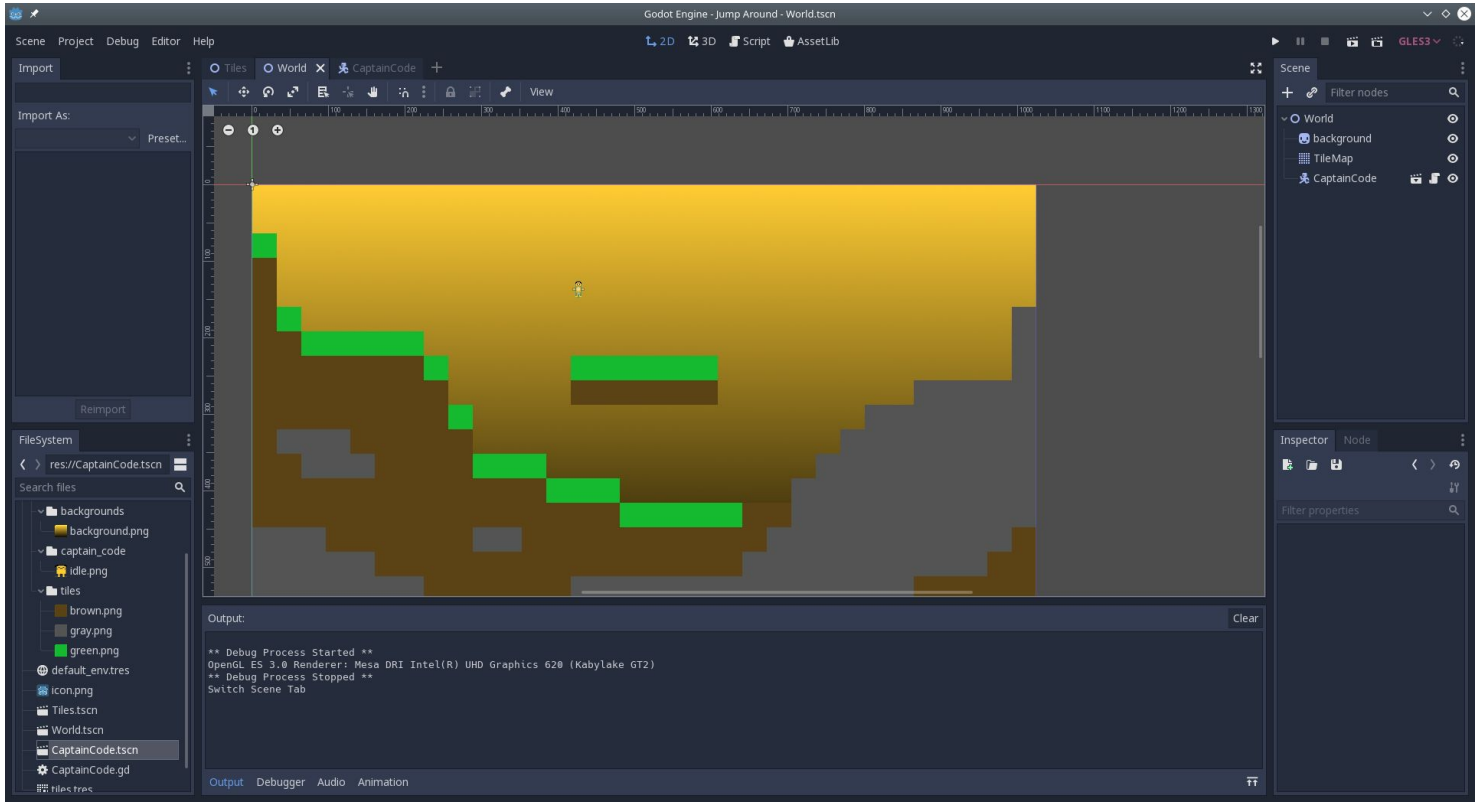
Importing




Scene
Project
Debug
Editor
Help
Import
Tiles
World
idle.png
Import As:
Texture
Preset...
Stream
0
Size Limit
0
Detect 3d
On
Compress
Mode
Lossle
Normal Map
Detect
Flags
Repeat
Disabled
2D, Detect 3D
2D
2D Pixel
3D
Set as Default for 'Texture'



Enter into the World



Jump around!

```
extends KinematicBody2D

export (int) var speed = 240
export (int) var jump_speed = -320
export (int) var gravity = 600

var velocity = Vector2.ZERO

func get_input():
	velocity.x = 0
	if Input.is_action_pressed("ui_right"):
		velocity.x += speed
	if Input.is_action_pressed("ui_left"):
		velocity.x -= speed
```

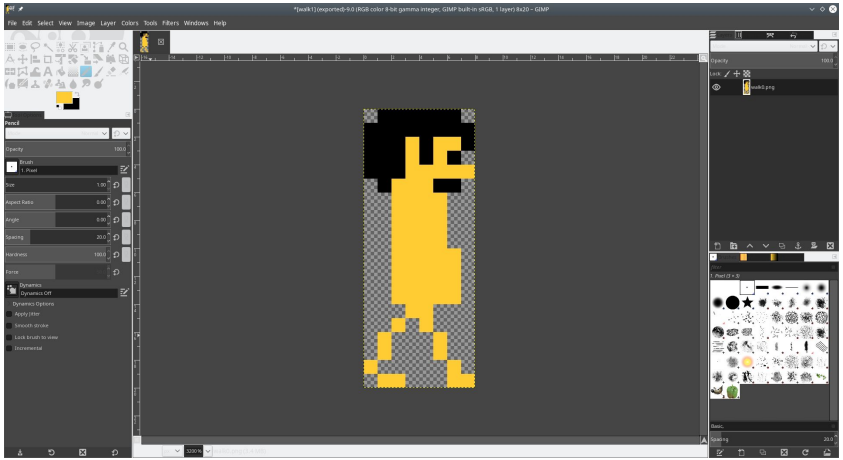
```
func _physics_process(delta):
	get_input()
	velocity.y += gravity * delta
	velocity = move_and_slide(velocity, Vector2.UP)
	if Input.is_action_just_pressed("jump"):
		if is_on_floor():
		velocity.y = jump_speed
```

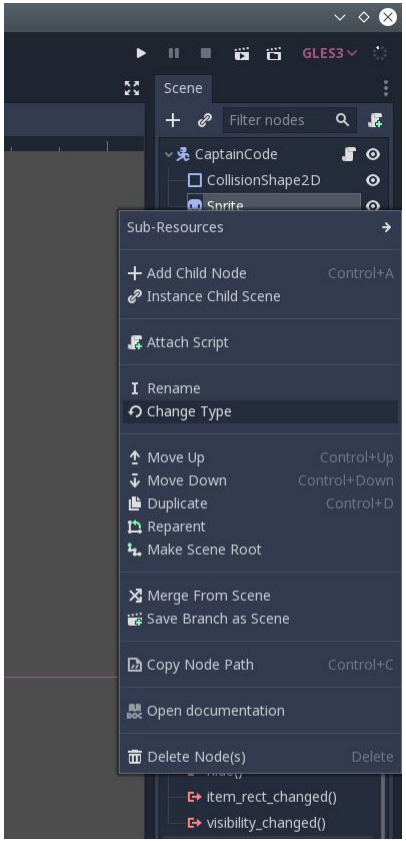
http://kidscancode.org/godot_recipes/2d/platform_character/

What is an input?

Animations!

- Change Captain Code from Sprite to AnimatedSprite
- New SpriteFrames in Frames
- Add the following:
  - Idle
  - Run
  - Jump





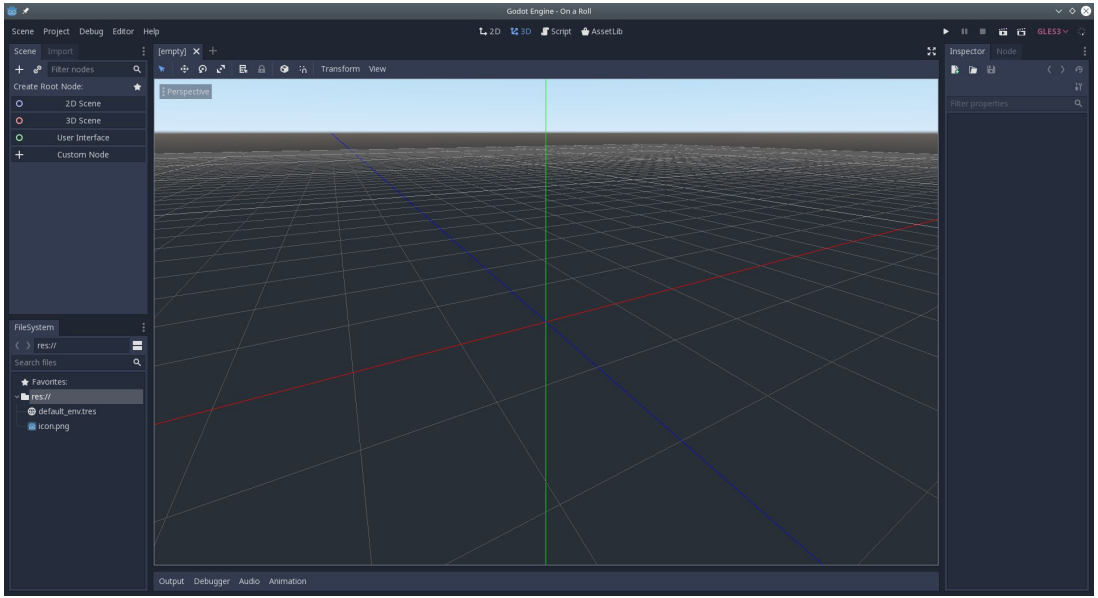
Play the right animation

```
func get_input() -> int:
	var res : int = 0
	velocity.x = 0
	if Input.is_action_pressed("ui_right"):
		velocity.x += speed
		res = 1
	if Input.is_action_pressed("ui_left"):
		velocity.x -= speed
		res = -1
	return res
```

```
func _physics_process(delta):
	var dir := get_input()
	if dir > 0:
		$Sprite.flip_h = false
		$Sprite.play("walk")
	elif dir < 0:
		$Sprite.play("walk")
		$Sprite.flip_h = true
	else:
		$Sprite.play("idle")
…
	if is_on_floor():
		...
	else:
		$Sprite.play("jump")
```

What about 3D?

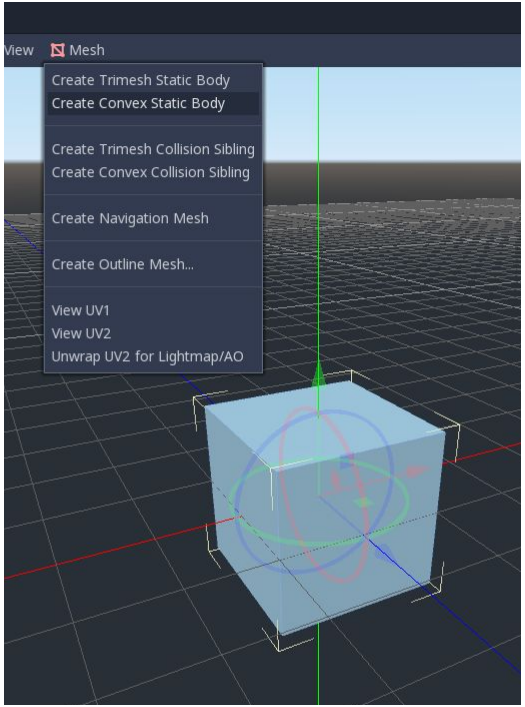
- You will learn about:
  - GridMaps and Mesh libraries
  - Basic lights and shadows
  - 3D collisions

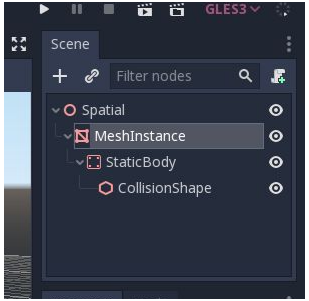


Meshes and Maps

- MeshInstance
  - Mesh: CubeMesh
  - Create Convex Static Body

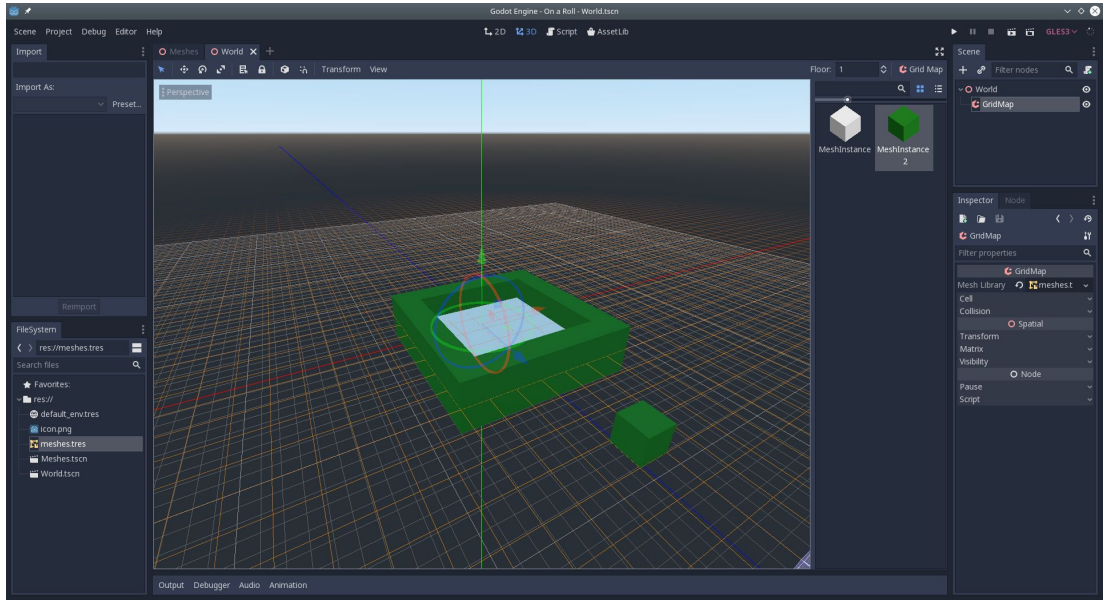
- Material
- SpatialMaterial
- Albedo





A World

- New 3D Scene - World
  - GridMap
  - Mesh Library = meshes.tres
- Don’t forget the Camera!



A movable sphere

- KinematicBody - Player
  - MeshInstance
  - CollisionShape

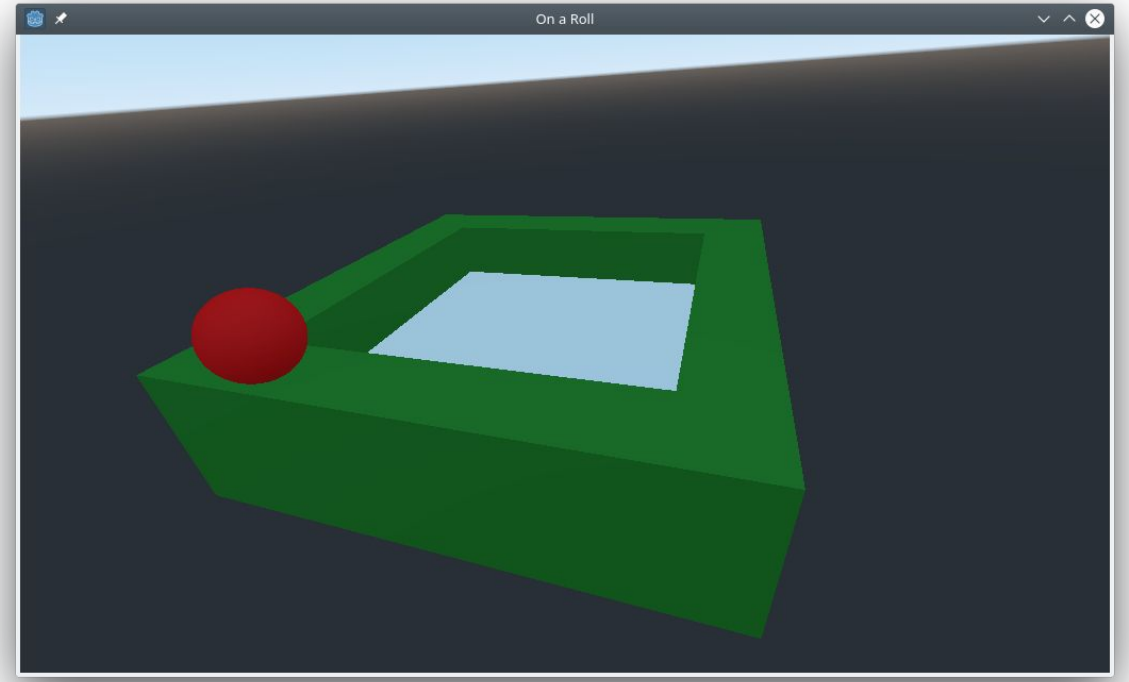
- Setup inputs

- Add to world

Movement script

```
extends KinematicBody

var velocity : Vector3 = Vector3.ZERO

const SPEED = 10
const GRAVITY = 10

func _physics_process(delta):
    velocity.y -= GRAVITY * delta
    if Input.is_action_pressed("forward"):
        velocity.z = -SPEED
    elif Input.is_action_pressed("back"):
        velocity.z = SPEED
    else:
        velocity.z = 0
if Input.is_action_pressed("left"):
    velocity.x = -SPEED
elif Input.is_action_pressed("right"):
    velocity.x = SPEED
else:
    velocity.x = 0

velocity = move_and_slide(velocity, Vector3.UP)
```

On a roll!

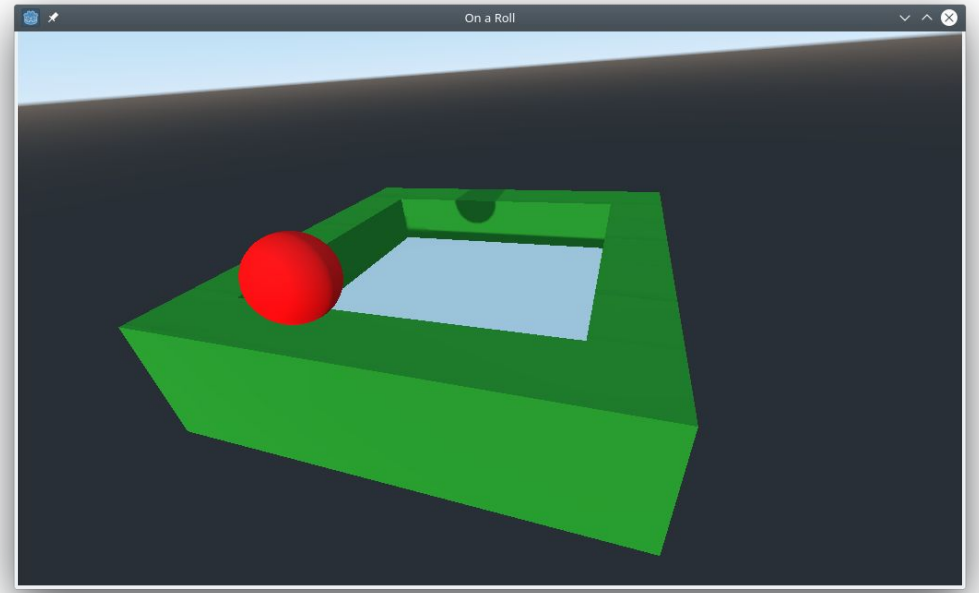


Various Bodies

- StaticBody
  - Static, does not move
  - Perfect for walls and such
- KinematicBody
  - Can be moved about
  - Commonly used for characters
- RigidBody
  - Simulates newtonian physics

Let’s add some light

- DirectionalLight
- Enable shadows



Let’s make the ball jump

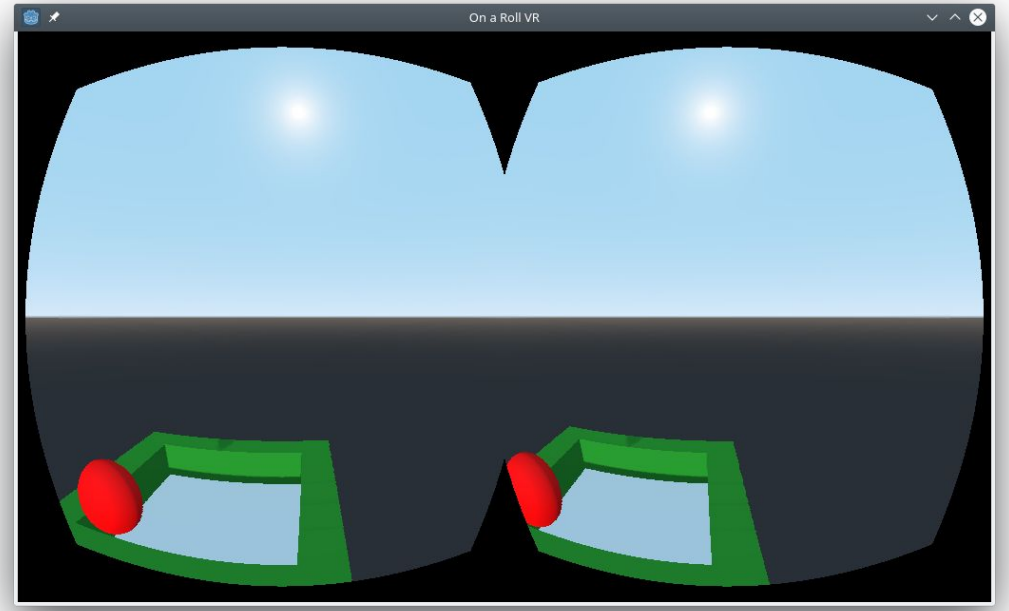
- Add input “jump”
- Add to _physics_process:

```
if is_on_floor():
if Input.is_action_just_pressed("jump"):
        velocity.y = JUMP
```

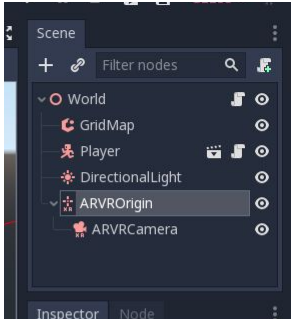
Going VR

- Godot comes with AR and VR support
- I’ve played around with SteamVR (Oculus DK2) and Google Cardboard

- Supports cameras and wands



Setting up VR



- Replace Camera node with ARVROrigin and ARVRCamera

- Initialize sub-system:

```
func _ready():
    var interface = ARVRServer.find_interface("Native mobile")
    if interface and interface.initialize():
        get_viewport().arvr = true
```

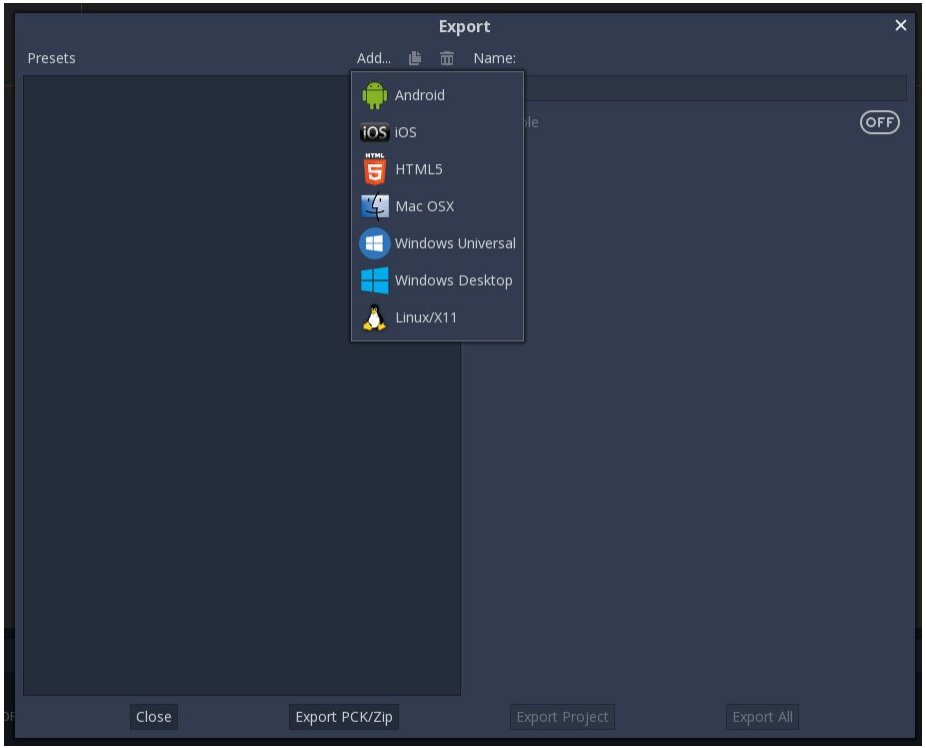
- Preferably do some error handling :-)

Try it!

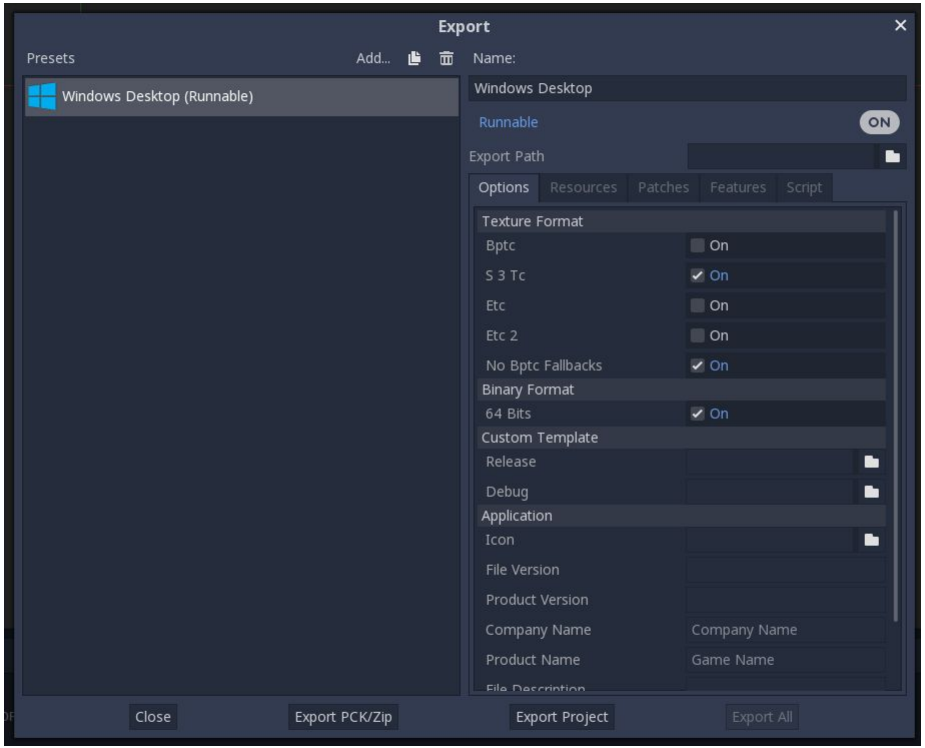
Deploying Godot projects

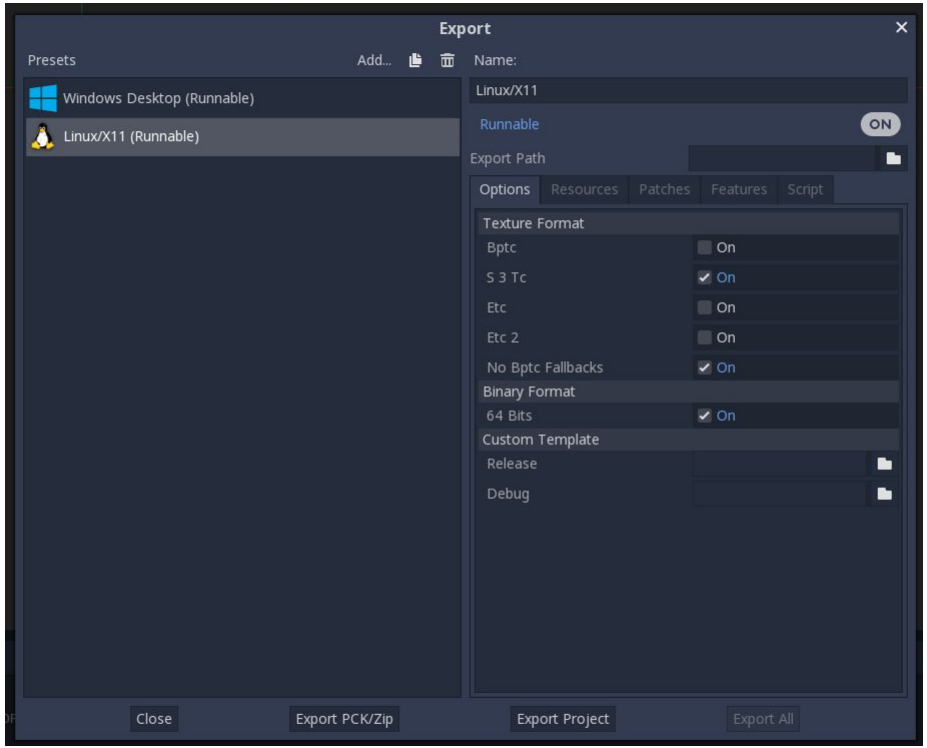
- Editor
  - Manage Export Templates
- Project
  - Export

- Requires some setup
  - Android
  - iOS
  - Windows Universal
- And you can build your own!

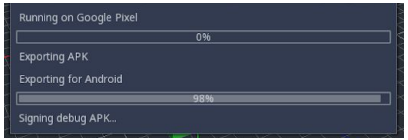


Deploying Godot projects





Android development



Things we didn’t learn

- Sound
- Dynamic instantiation
- 3D assets from Blender
- Skeleton animations
- Shaders
- Path finding
- C# bindings
- GDNative, i.e. rust, C++, and beyond
- Setting up an Android target
- Setting up an iOS target
- And much, much more...