Download Starting Project

1. Open bit.ly/2LSKbw0 in a browser to download "Crash-Course-Unity-2D-Assets.zip"

Crash Course Godot 3

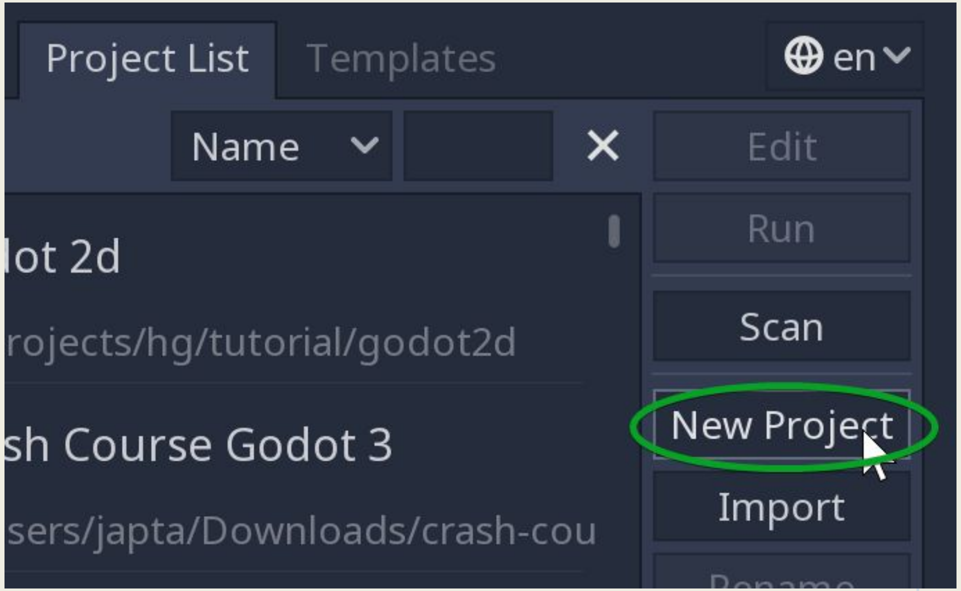
Building 2D interactive art in Godot 3

Goal

- Create an interactive 2D environment
- A brief start in GD scripting

Starting a new project

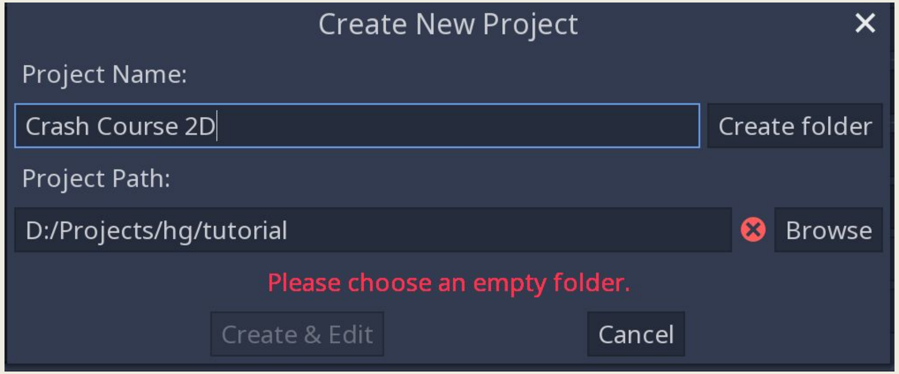
Step 1: Click “New Project”



Starting a new project

Step 2: Type in a project name and an empty folder.

Note: "Project Path" must be empty. Consider using the "Create folder" button to create a folder w/ same name as project name.



Create New Project

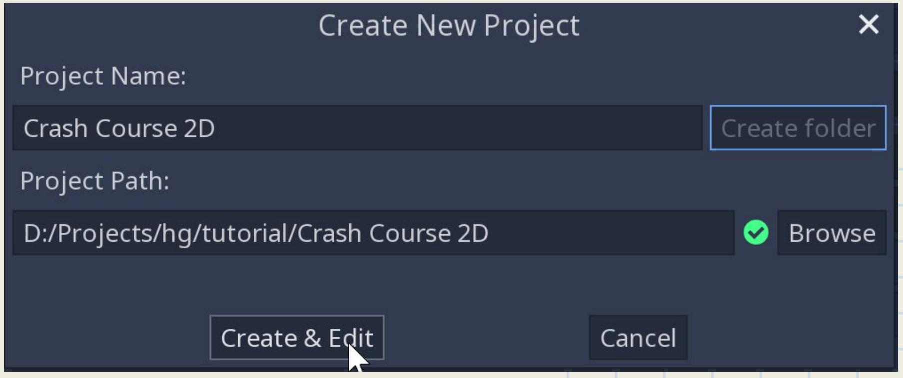
Project Name:

Crash Course 2D

Project Path:

D:/Projects/hg/tutorial

Please choose an empty folder.



Create New Project

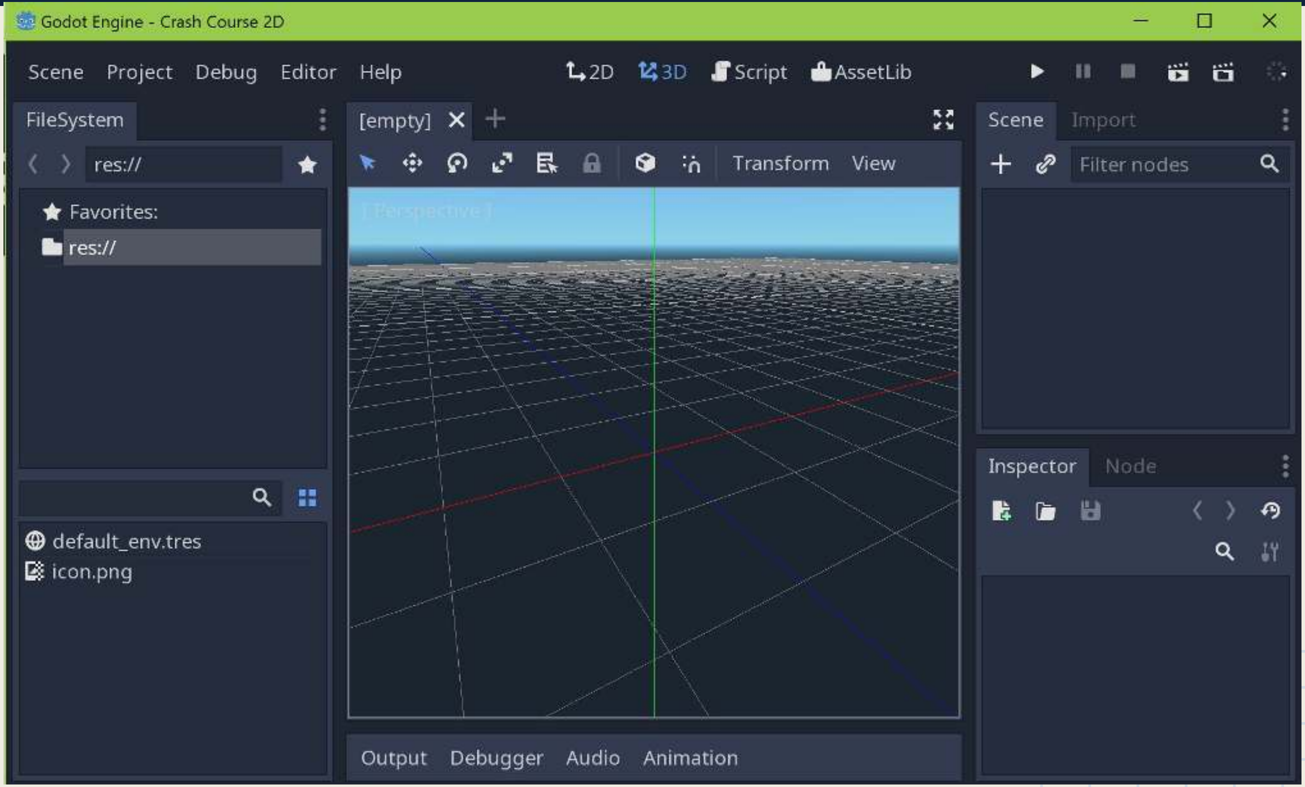
Project Name:

Crash Course 2D

Project Path:

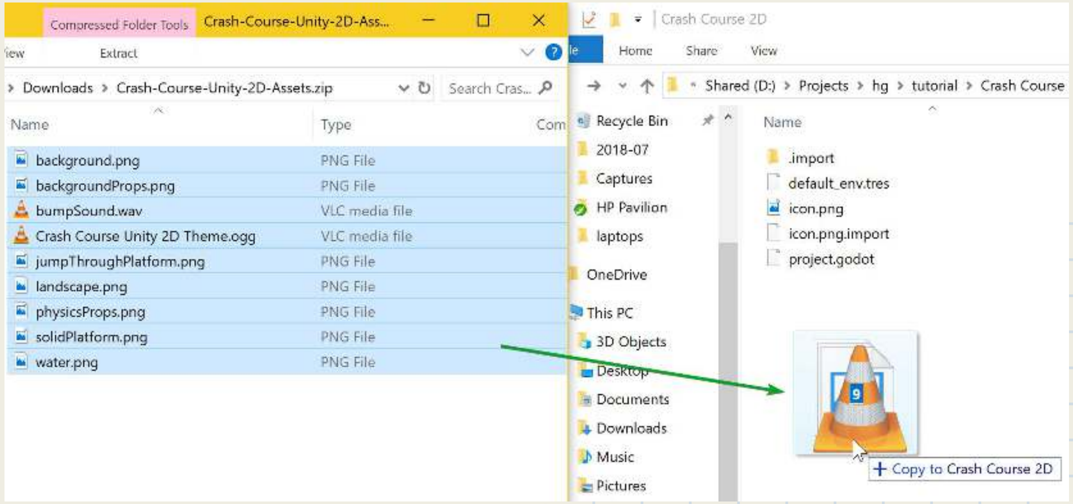
D:/Projects/hg/tutorial/Crash Course 2D

Making a game



Importing stuff

1. Open the new project's folder
2. Copy the contents of "Crash-Course-Unity-2D-Assets.zip" to this folder.
3. Switch to Godot.

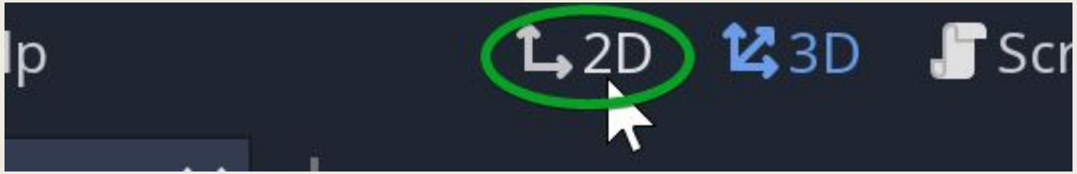


Name	Type
background.png	PNG File
backgroundProps.png	PNG File
bumpSound.wav	VLC media file
Crash Course Unity 2D Theme.ogg	VLC media file
jumpThroughPlatform.png	PNG File
landscape.png	PNG File
physicsProps.png	PNG File
solidPlatform.png	PNG File
water.png	PNG File

Name
.import
default_env.tres
icon.png
icon.png.import
project.godot

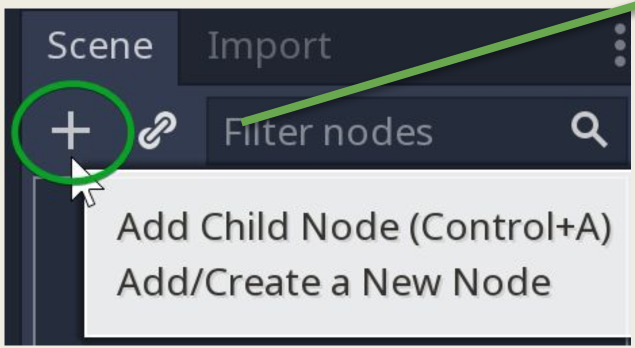
Scene View

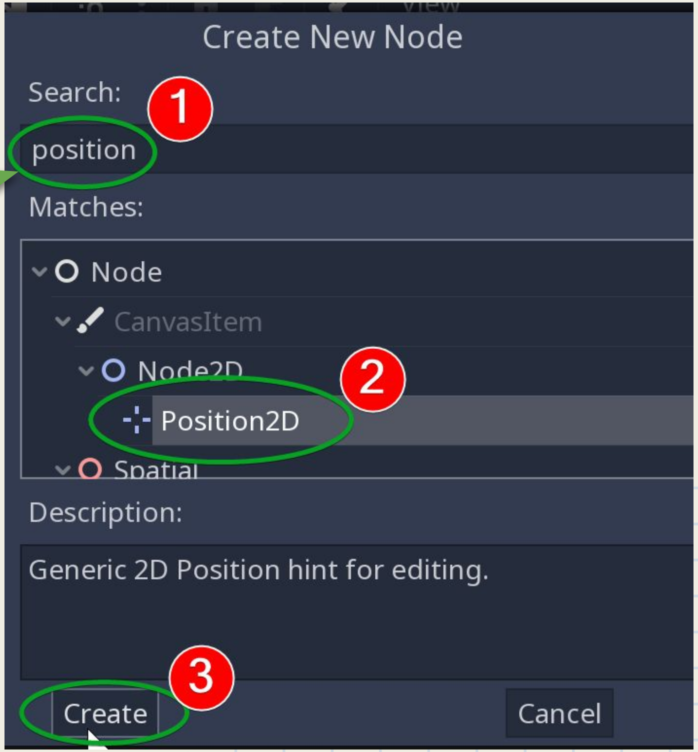
- Near the top of the screen, click on “2D” to toggle to the 2D scene view.



Scene View

- Create a Position2D node in the Scene pane.



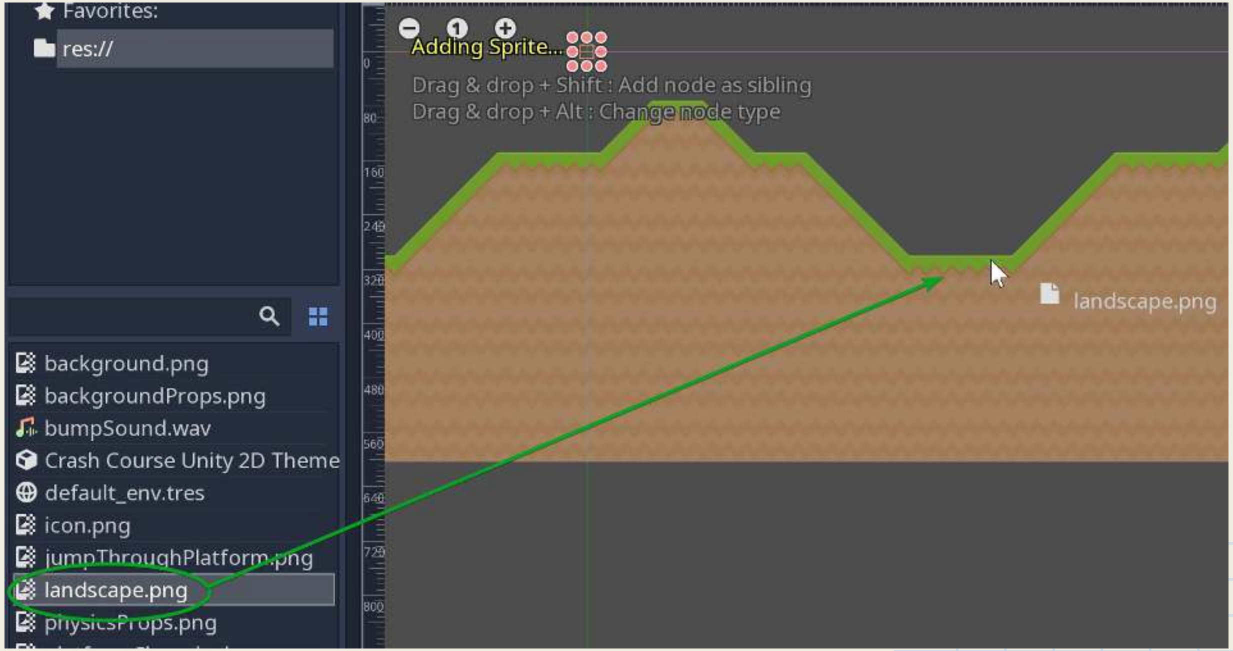


Generic 2D Position hint for editing.

- Save this scene!

Add a sprite to a new scene

1. Drag & Drop the landscape.png to the scene view.



Drag & drop + Shift : Add node as sibling
Drag & drop + Alt : Change node type

Navigating the scene in 2D

- 3-button mouse:
  - Left-click to select objects
  - Click and hold on the scroll wheel to pan
  - Scroll wheel to zoom in and out

Add a sprite to a new scene

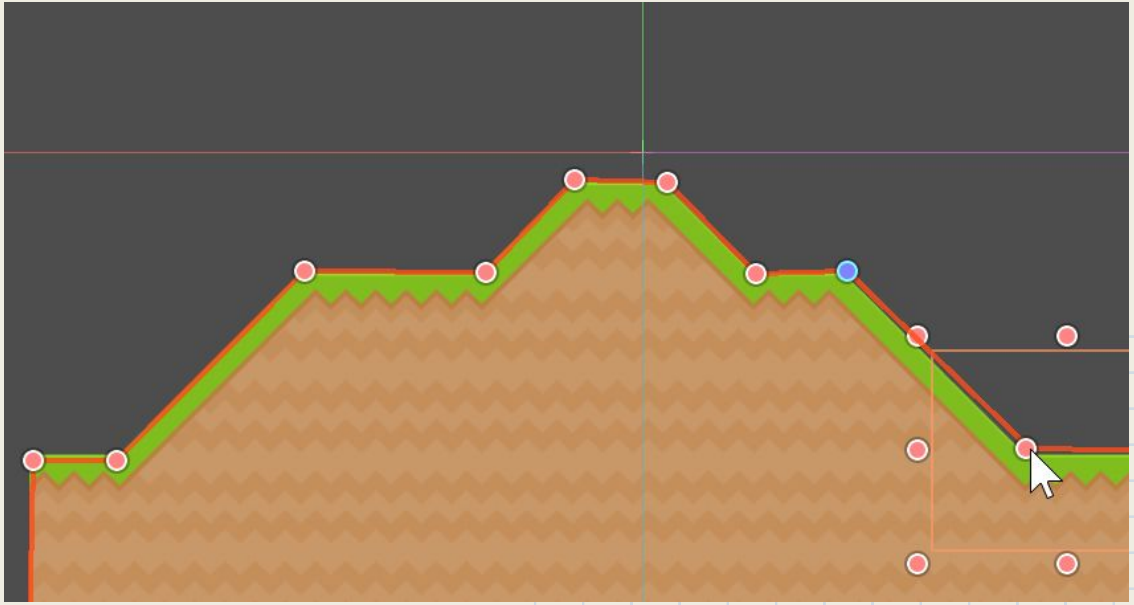
1. Move the landscape to the bottom-right, below the origin axis.
2. Play the game (accept adding the scene as the first scene to play).
3. Notice that, without a camera,
   Godot defaults (0, 0) at the top-left of the screen.

Add 2D collision

1. In Scene Pane, add a StaticBody2D to the root node.
2. On the StaticBody2D node, add a CollisionPolygon2D.

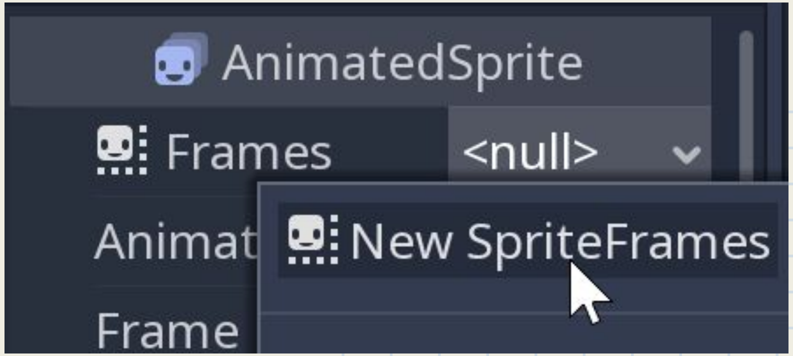
Add 2D collision

1. While having the CollisionPolygon2D node is highlighted, click on the Scene view to create a polygon.



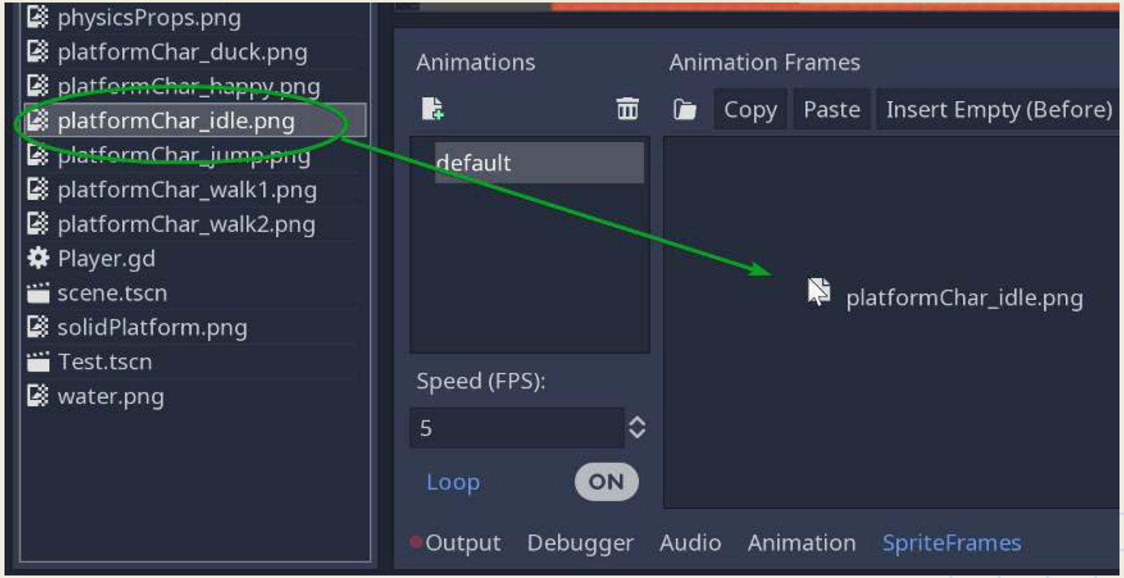
Add a player

1. Right-click the Position2D again, and create a new child node, KinematicBody2D.
2. Right-click KinematicBody2D, then add AnimatedSprite.
3. In properties, create a new SpriteFrames.
4. Click on SpriteFrames.



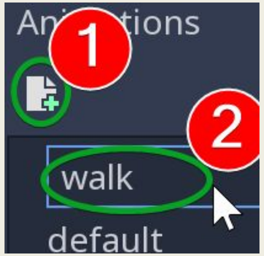
Add a player

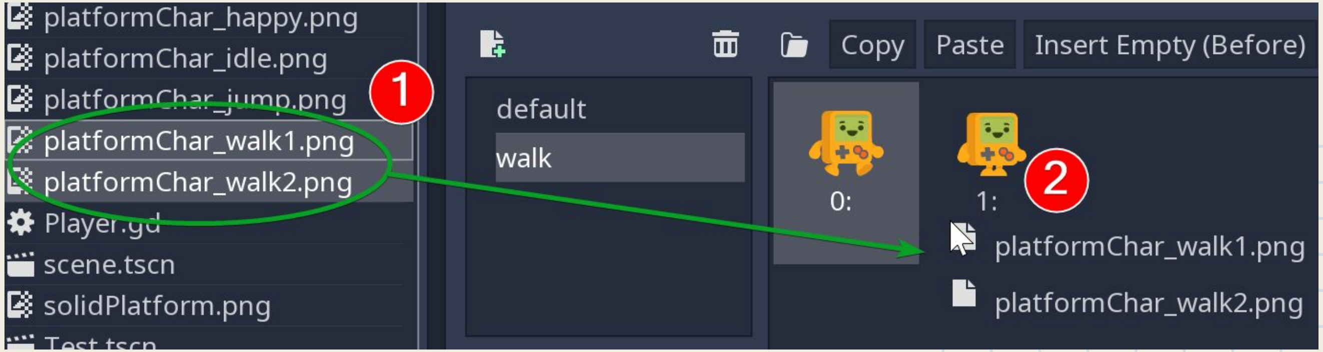
1. Drag the platformChar_idle.png into the bottom-pane, Animation Frames.



Add a player

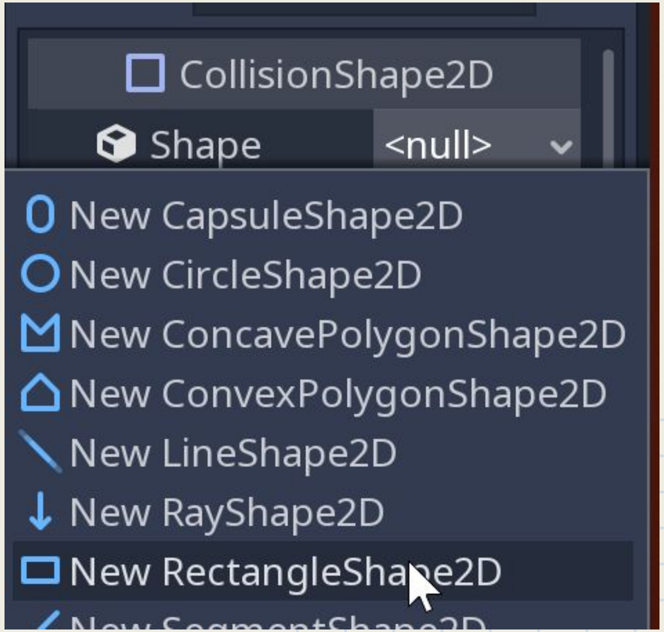
1. Create a new animation, name it "walk."
2. Drag and drop the 2 platformChar_walk#.png sprites into the animation window to create a new animation.





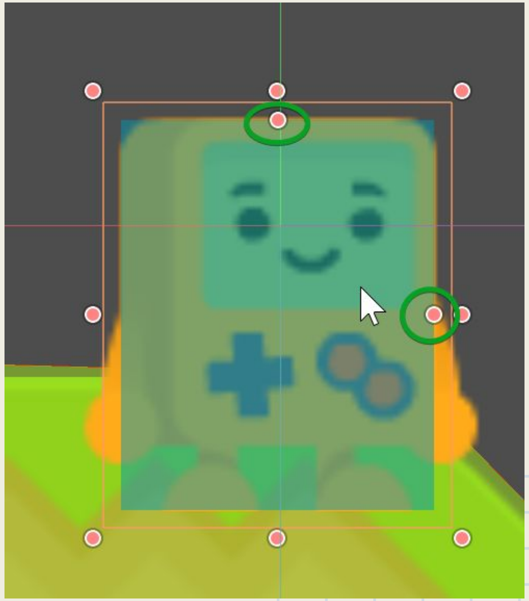
Add controls

1. Right-click the KinematicBody2D, and create a new CollisionShape2D.
2. In Properties, create a new RectangleShape2D.



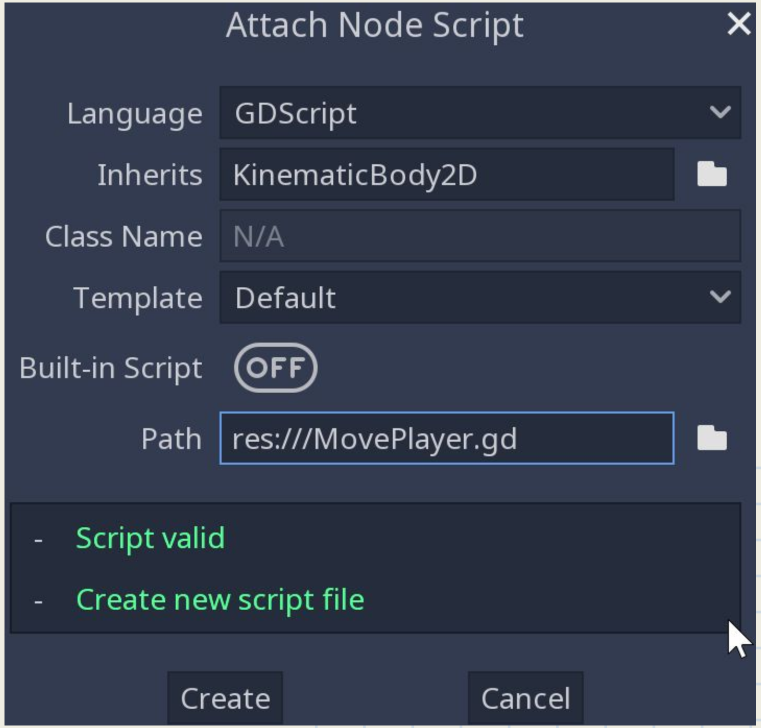
Add controls

1. Adjust the size of the box collider with the inner 2 circle nodes.



Add controls

1. Select KinematicBody2D, and create a new script, “MovePlayer.gd”



Language GDScript

Inherits KinematicBody2D

Class Name N/A

Template Default

Built-in Script OFF

Path res:///MovePlayer.gd

- Script valid
- Create new script file

Check the Script!

```
extends KinematicBody2D

# class member variables go here,
for example:
# var a = 2
# var b = "textvar"

func _ready():
	# Called when the node is added
to the scene for the first time.
	# Initialization here
	pass
```

```
#func _process(delta):
#	# Called every frame. Delta is
time since last frame.
#	# Update game logic here.
#	pass
```

GD Script Syntax

- Extending a class:
  - `extends KinematicBody2D`
- Declaring a variable:
  - `var a = 2`
  - `var b = "textvar"`
  - `var vec = Vector2(0, 0)`
  - `var arr = [1, 2, 3]`
  - `var dict = {"key": "value", 2: 3}`
- Declaring constants:
  - `const INT_A = 2`
  - `const STRING_B = "textvar"`

GD Script Syntax

- Defining a new function:
  ```
  func custom_function():
      pass
  ```
- Conditionals:
  ```
  if true:
      pass
  elif true:
      pass
  else:
      pass
  ```

GD Script Syntax

- While loop:
  ```
  ○ while(false):
  ○     pass
  ```
- For loop:
  ```
  ○ for i in range(20):
  ○     print(i)
  ```
- Return:
  ```
  ○ return false
  ```
- Enum:
  ```
  ○ enum Version {ALPHA, BETA = 10}
  ○ Version.ALPHA
  ```

GD Script Syntax

- Comments:
  - `# This is a comment`
- Inner class:
  - `class Test:`
  - `    var value = 1`
- Override Methods:
  - `# Init is the Constructor`
  - `func _init():`
  - `    # Call base method`
  - `    ._init()`
  - `    # Create object of inner class, Test`
  - `    var lv = Test.new()`

Methods to Override

- _ready() is called when the game begins.
  - `func _ready():`
  - `    pass`
  - Basically Unity equivalent of “Start().”
- _process(float) is called on each frame.
  - `func _process(delta):`
  - `    pass`
  - Parameter “delta” is a fraction of a second (float).
  - Basically Unity equivalent of “Update().”

Methods to Override

- _physics_process(float) is called on each Physics step.
  - `func _physics_process(delta):`
  - `    pass`
  - Parameter “delta” is a fraction of a second (float).
  - Basically Unity equivalent of “FixedUpdate().”

Write a Script

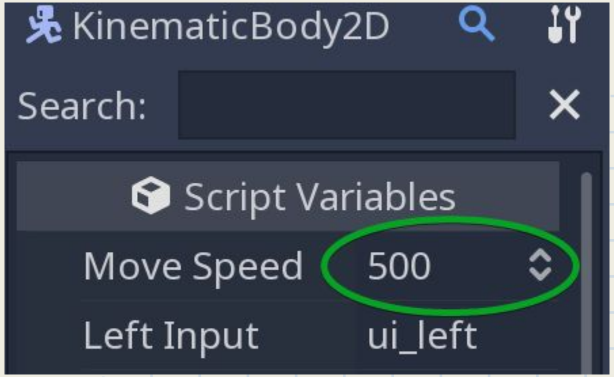
```
extends KinematicBody2D

export(float) var move_speed = 500
export(String) var left_input = "ui_left"
export(String) var right_input = "ui_right"
export(String) var up_input = "ui_up"
export(String) var down_input = "ui_down"

var input = Vector2(0, 0)


# More code on the next slide!
```

Write a Script

```
func _physics_process(delta):
    input.x = 0
    if Input.is_action_pressed(left_input):
        input.x -= move_speed
    if Input.is_action_pressed(right_input):
        input.x += move_speed
    input.y = 0
    if Input.is_action_pressed(up_input):
        input.y -= move_speed
    if Input.is_action_pressed(down_input):
        input.y += move_speed
    move_and_slide(input)
```

Adjusting Script Numbers

1. Play the game! Use arrow keys.
2. Close the game, and look at Properties.
3. Notice the following line makes variables exposed to the Properties pane:
   a. `export(float) var move_speed = 500`



KinematicBody2D

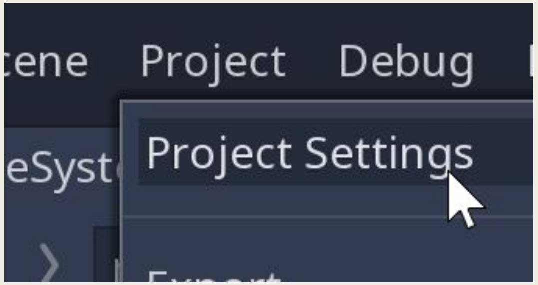
Search:

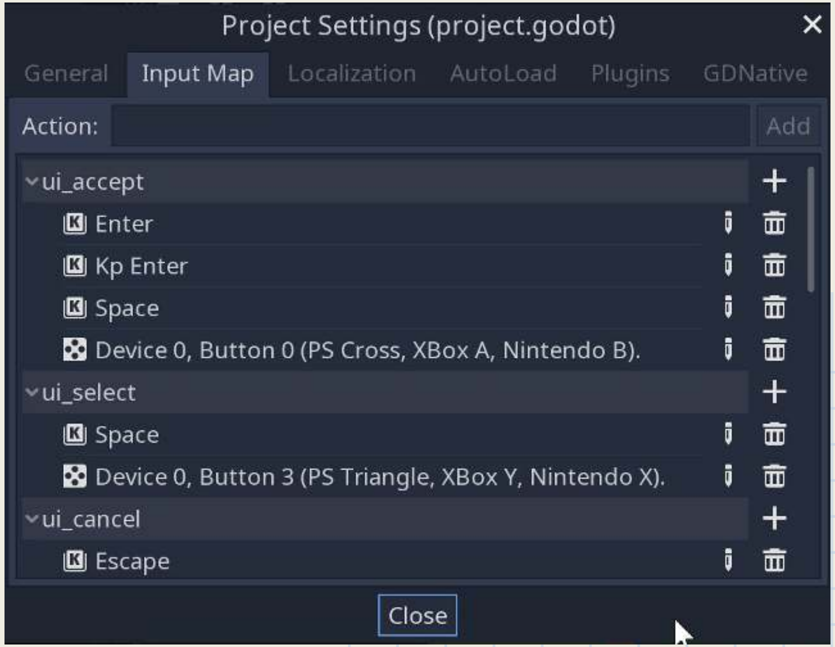
Script Variables

Move Speed	500
Left Input	ui_left

Adjusting Input

1. In the menu bar, select "Project -> Project Settings."
2. Click the "Input Map" tab.
3. Observe entries "ui_up," "ui_down," etc.





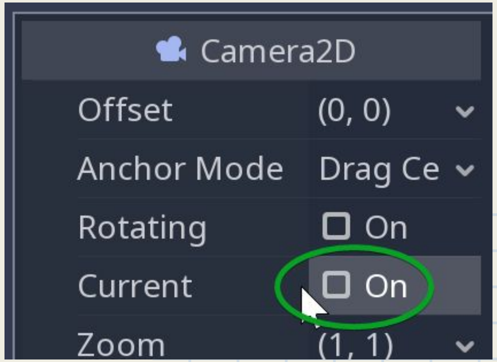
ui_accept
- Enter
- Kp Enter
- Space
- Device 0, Button 0 (PS Cross, XBox A, Nintendo B).

ui_select
- Space
- Device 0, Button 3 (PS Triangle, XBox Y, Nintendo X).

ui_cancel
- Escape

Following Camera

1. Right-click the KinematicBody2D, and create a new Camera2D.
2. Make sure the “Current” checkbox is checked in the Properties, or else the camera won’t work.
3. Play the game!

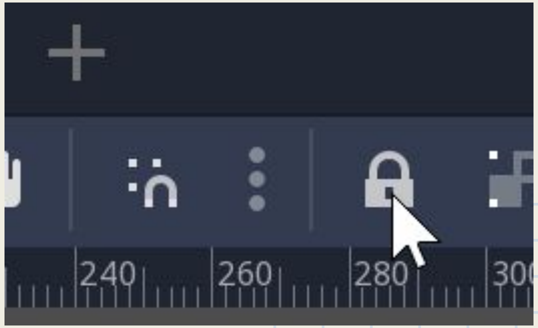


Camera2D

Offset	(0, 0)
Anchor Mode	Drag Ce
Rotating	☐ On
Current	☐ On
Zoom	(1, 1)

As an Aside

1. Hard to move KinematicBody2D?
2. Select AnimatedSprite2D.
3. In the Scene view, click the "toggle lock" button.
4. Do the same thing for CollisionShape2D.
5. This will make both nodes unselectable.



Update Animation

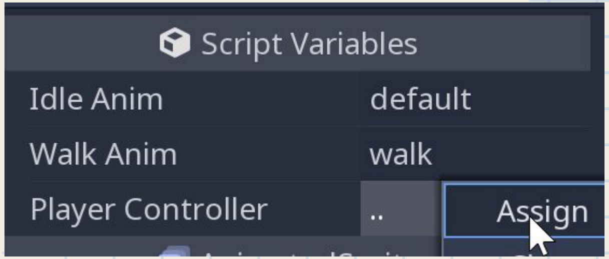
1. Select AnimatedSprite.
2. Create a script on the AnimatedSprite.

Update Animation

```
extends AnimatedSprite

export(String) var idle_anim = "default"
export(String) var walk_anim = "walk"
export(NodePath) var player_controller = ".."

func _process(delta):
    var node = get_node(player_controller)
    if (node.input.x != 0) || (node.input.y != 0):
        play(walk_anim)
    else:
        play(idle_anim)
```

Update Animation

1. Note the following 2 lines:
   a. `export(NodePath) var player_controller = ".."`
   b. `var node = get_node(player_controller)`
2. get_node() retrieves the node (and the script attached to it!) based on a string path.
3. export(NodePath) provides a unique property that are assigned to nodes specifically.
4. Note that string is based on Unix directory standard, e.g. ".." = parent directory.


Script Variables

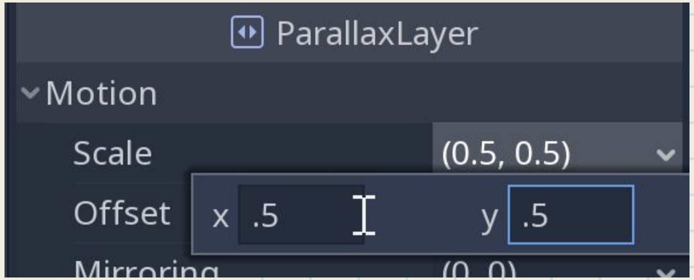
Idle Anim	default
Walk Anim	walk
Player Controller	..

Update Animation

1. Play the game! Use arrow keys.
2. Did the animation change?

Adding Background

1. Right-click Position2D, and create a ParallaxBackground node.
2. Right-click ParallaxBackground, and create a ParallaxLayer node.
3. Update the properties, and change the scale.


ParallaxLayer

Motion

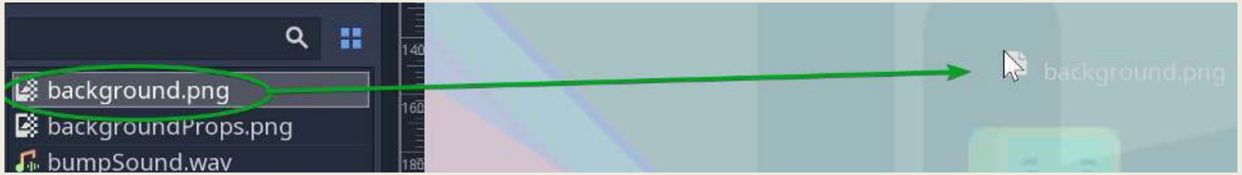
Scale (0.5, 0.5)

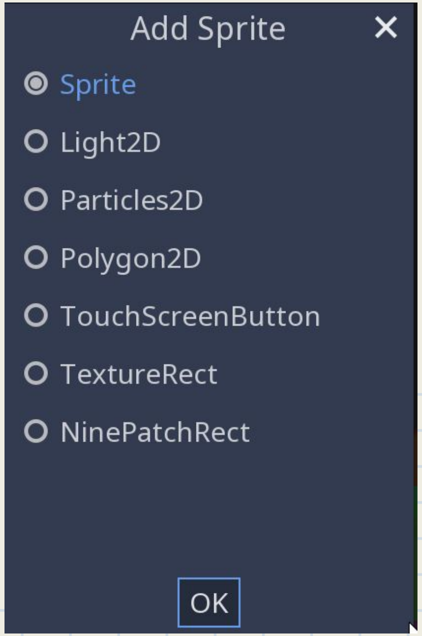
Offset x .5 y .5

Mirroring (0, 0)

Adding Background

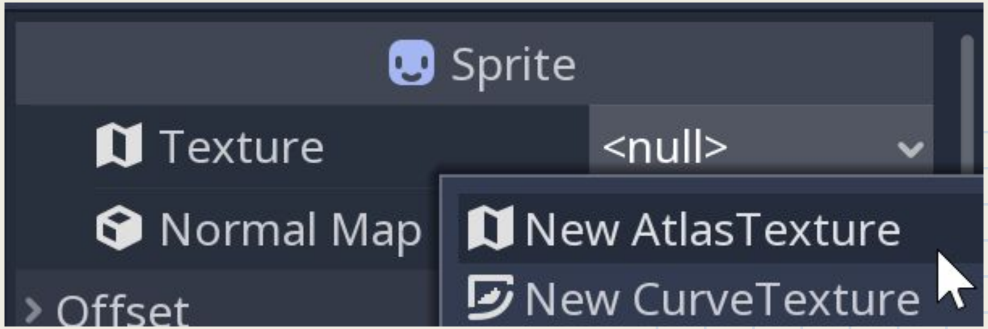
1. While ParallaxLayer is still highlighted, drag & drop background.png.
2. If "Add Sprite" dialog comes up, change to a Sprite.
3. Play the game!





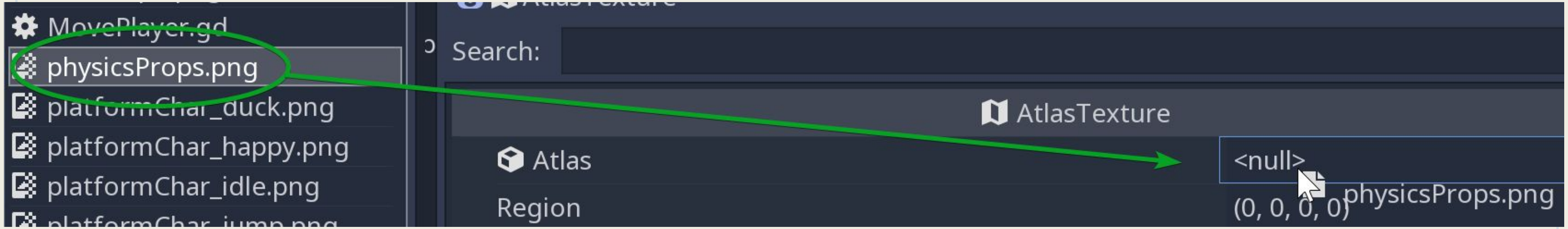
Add Rigidbodies

1. Right-click Position2D node, and create a Rigidbody2D.
2. Right-click the Rigidbody2D, and create a new Sprite.
3. In the properties, under “Texture,” select “New AtlasTexture.”
4. Click this property.



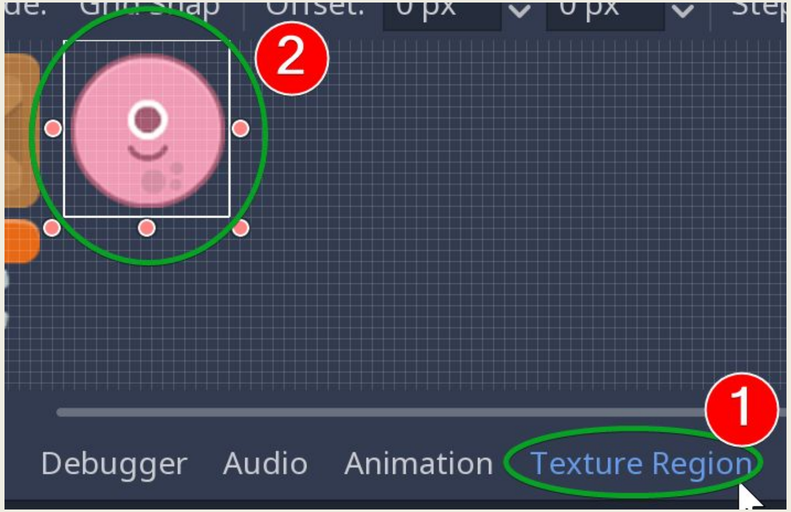
Add Rigidbodies

1. Drag the physicsProps.png in the Atlas property.



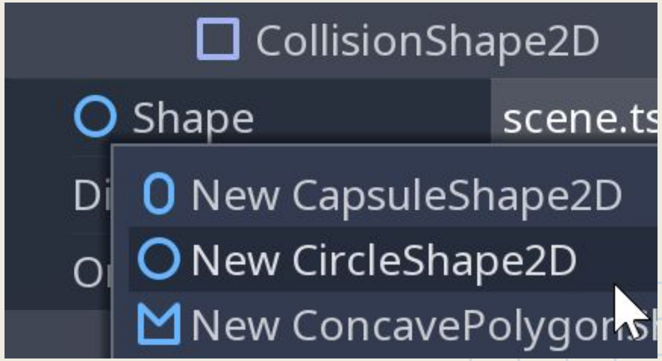
Add Rigidbodies

1. Click “Texture Region” at the bottom of the editor.
2. Click, hold, and drag over the circle to select its region.



Add Rigidbodies

1. Right-click the Rigidbody2D, and create a new CollisionSprite2D.
2. Assign a circle shape, and have it encompass the sprite.
3. Play the game!



Collision Detection Support?

1. Same as 3D; select Node pane, and assign a signal to a script!
2. This will create a function.

Collision Detection Support?

1. Instancing a scene:
   a. `var scene = load("res://scenes/MyScene.tscn")`
   b. `var scene_instance = scene.instance()`
   c. `scene_instance.set_name("scene")`
   d. `add_child(scene_instance)`