

## **Quick Platformer Guide**

*By AJ*

Construct 3 to Godot 4 conversion for tutorial

<https://www.construct.net/en/tutorials/platformer-game-2329>

**Layout/Background:**

**Tilemapping:**

**Layering: Collision Masks**

**Sprites and Collisions:**

- How to create a sprite

**Creating Platforms:**

- Tilemap

**Scripts & Player Movement:**

**Sprite Animations:**

**Enemy Sprite Creation:**

**Enemy-Player Collisions & Scripts:**

- Jumping on Enemy → Enemy Despawn
- Enemy Collides with Player → Player flashes(Player Despawn)
- Enemy Behavior:
  - Move left and right
  - Add 'edge' markers (Method 1) - Area2D
  - (Optional)Use raycasting and 'ledge' checkers

**UI:**

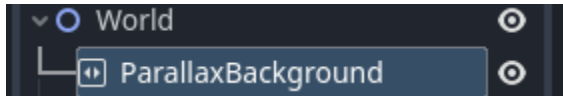
- Add Pickups counter

**Pickups Behavior and Script**

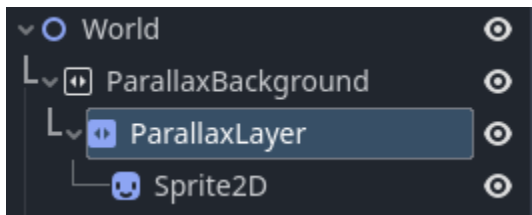
- Specific path (Node2D Path)
- Script:
  - Add to Score

### **Starting Out:**

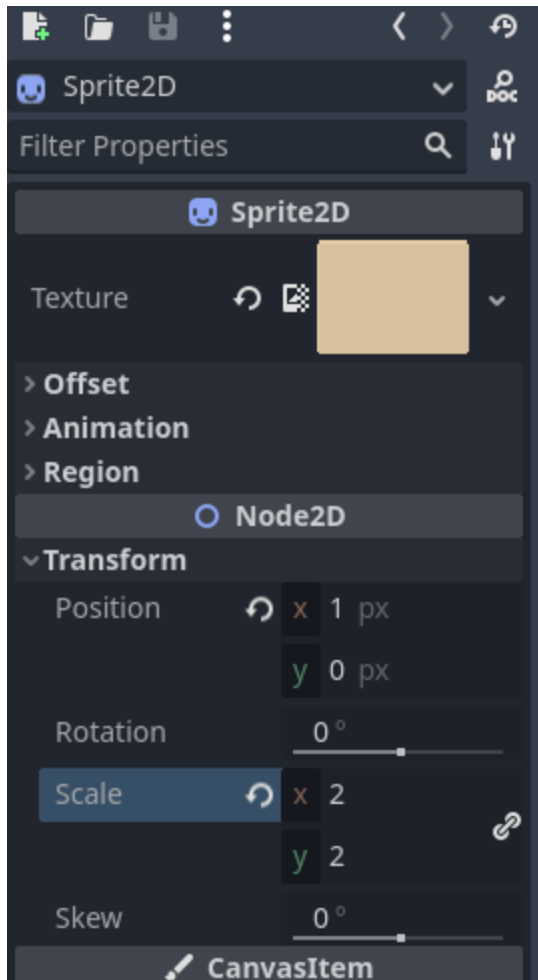
- Create a new project and then use a 2D node, name it 'World' and save the scene
- We will use this scene as our level
- We will start out with a background
- We're going to be making a parallax background



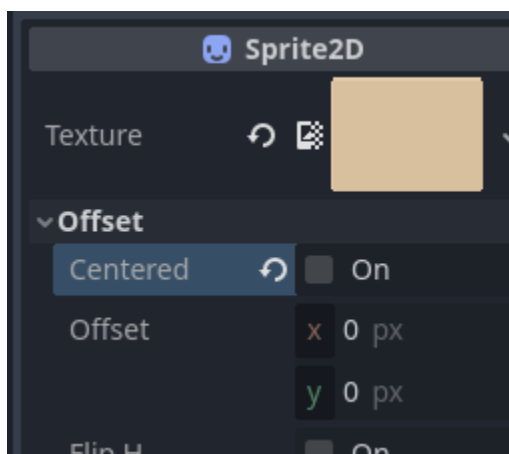
- Then add a parallaxlayer as a child of the background, and a sprite2d node as a child of that



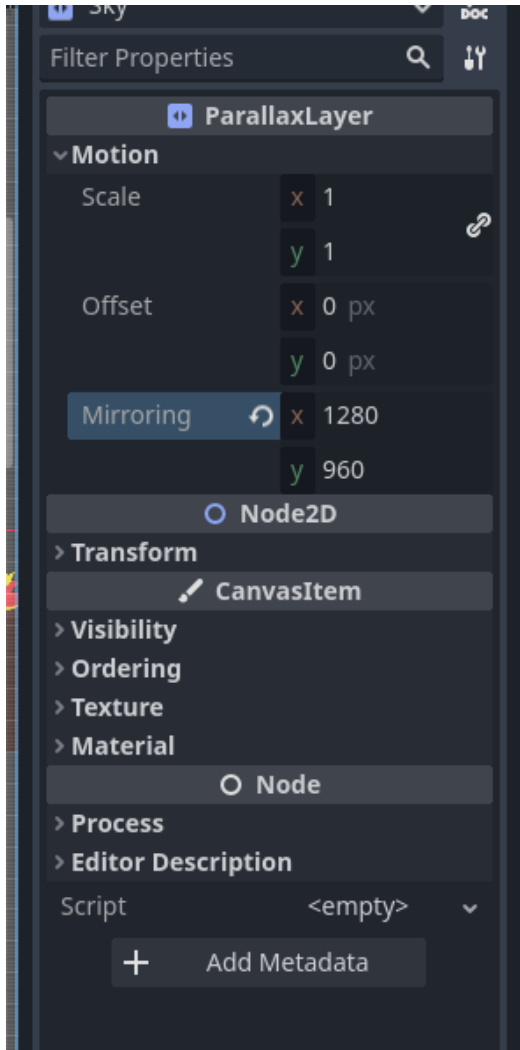
- Add the texture to the sprite2D and scale it



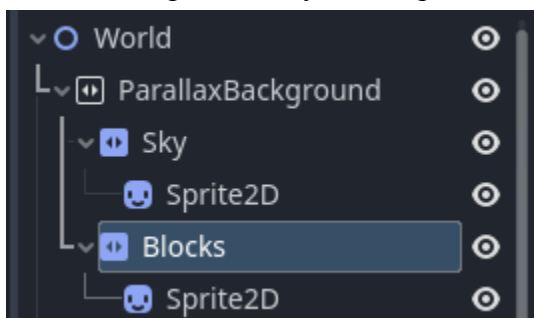
- 
- Then turn off the offset-> centered box



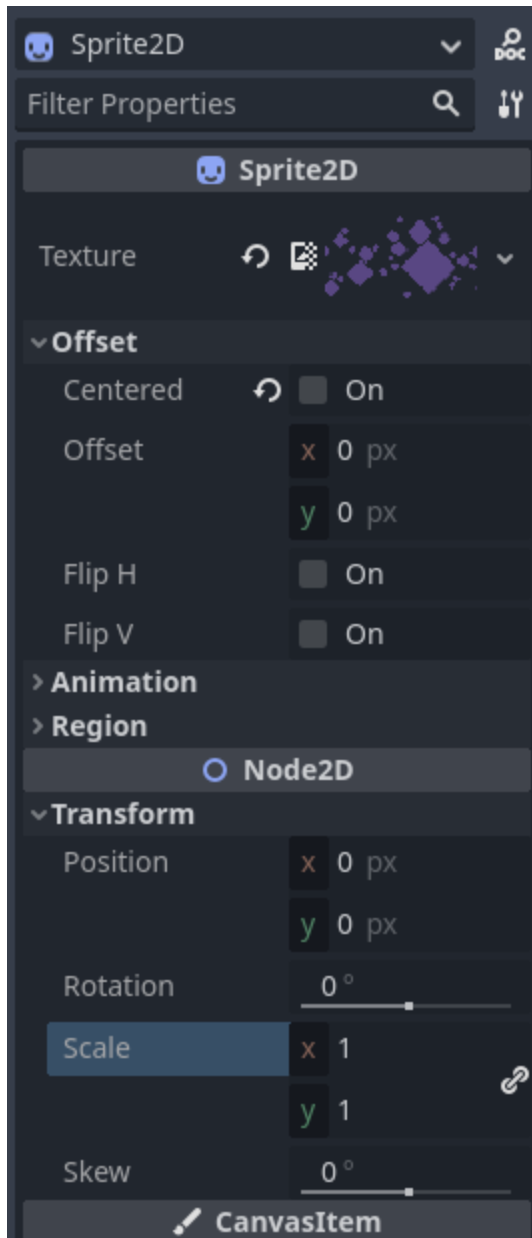
- 
- Now under parallax layer go to motion and turn on mirroring, stating the the pixels width and height for background image



- 
- Add another parallax layer and sprite2D



- 
- Repeat the process of turning off the centeredbox



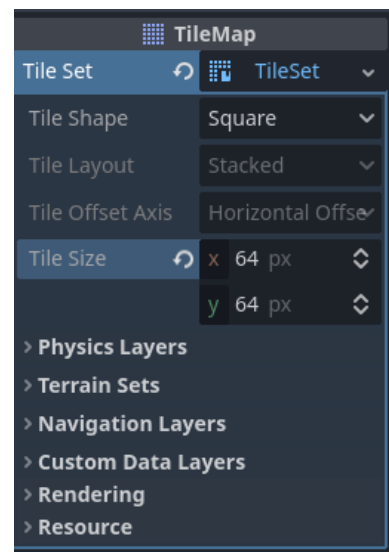
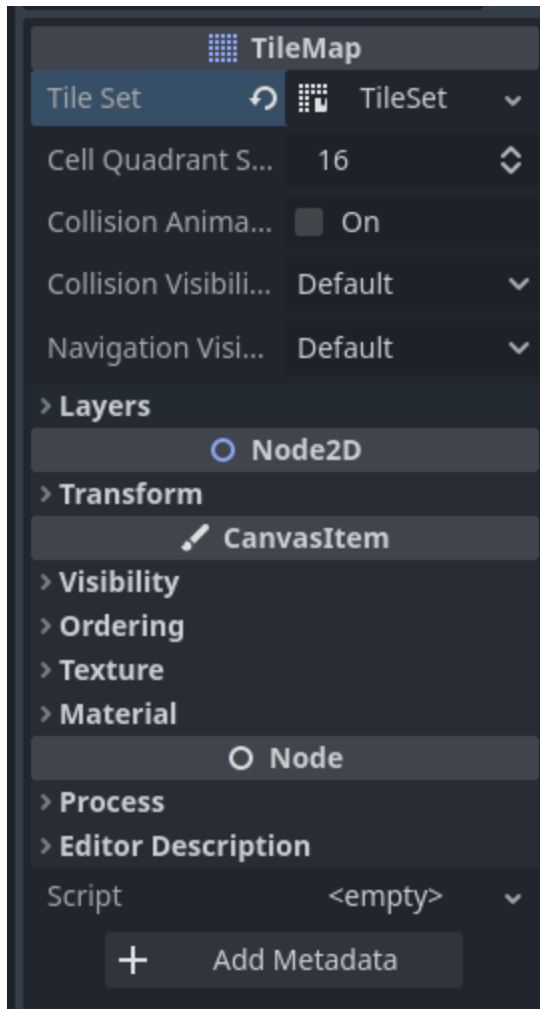
- 
- Edit the parallaxlayer, adding mirroring



- 
- We now have a repeating background!

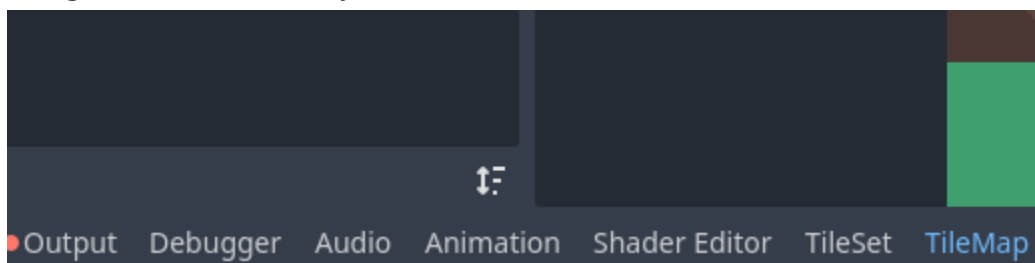
### **Tilemap:**

- Create a new scene, go to tilemap node, save scene as tilemap.
- Open *Tilesheet\tilesheet\_complete.png*
- Select tilemap and create a new tileset

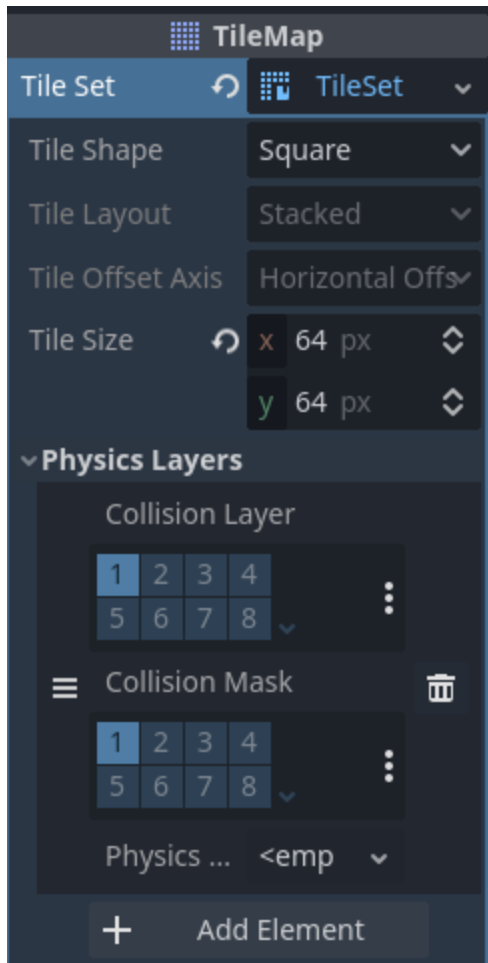


- On the right, change tilesize to 64 by 64 for this tileset

- Navigate to the bottom of your screen and select tileset

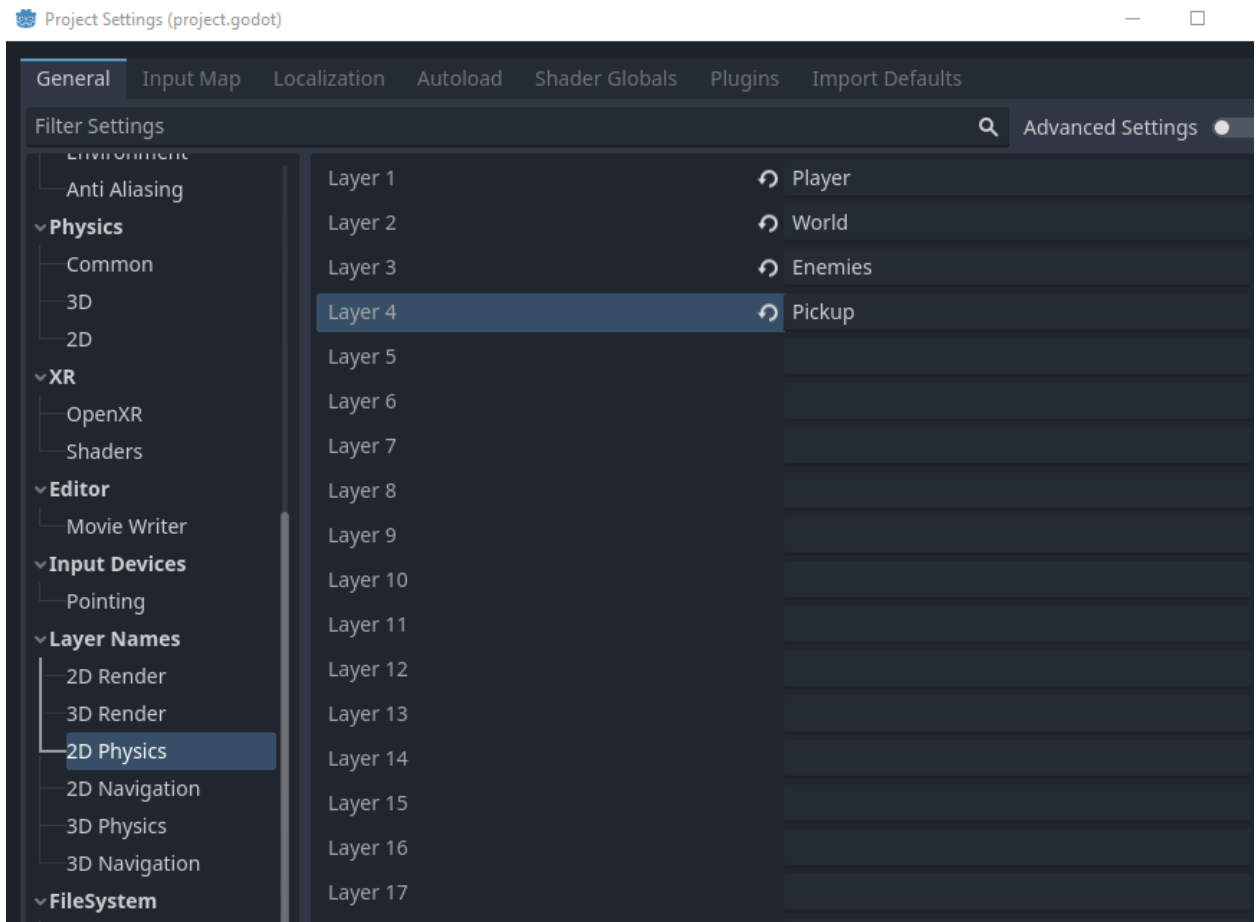


- Now import the tilesheet\_complete
- Use the eraser tool to erase existing tiles, and then deselect the eraser
- Now hold shift and then highlight over 4x4 areas to make it one single tile.
- Repeat the step for how many blocks you want to be included in the tilemap
- To add collisions to your tilemap tiles:
- First, go to the right and add a physics layer element

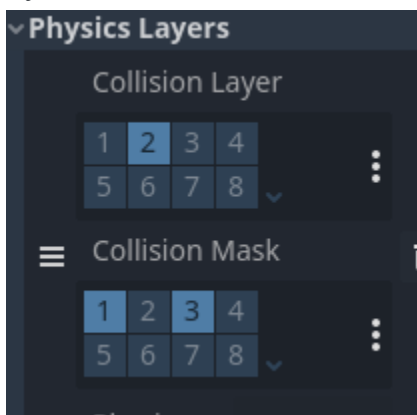


- 
- Collision layers dictate what layer the collision box will reside in, while the mask will dictate what layer it should look for and interact with.
- You can go to project settings and navigate to '2D Physics' to name the layers.

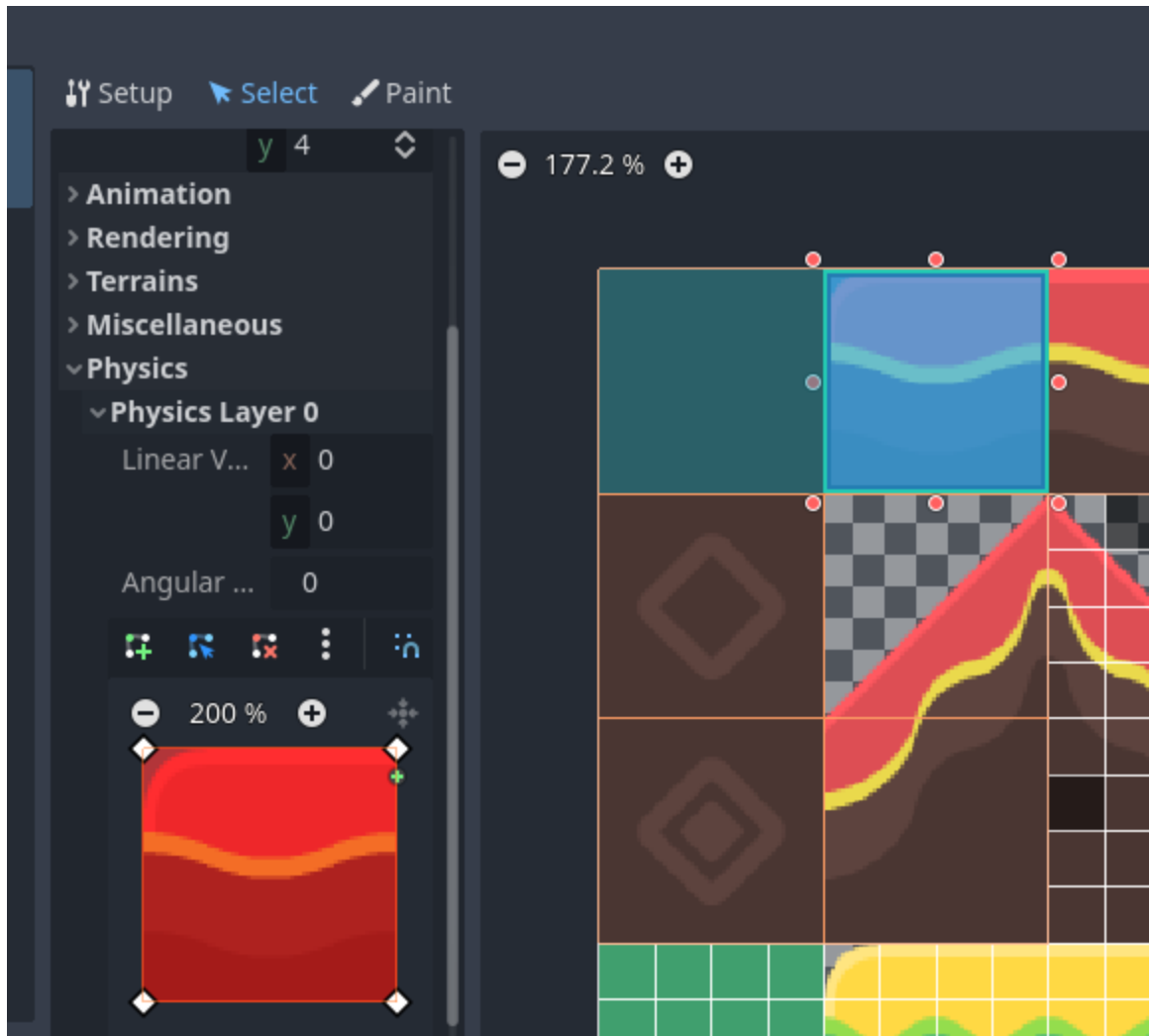




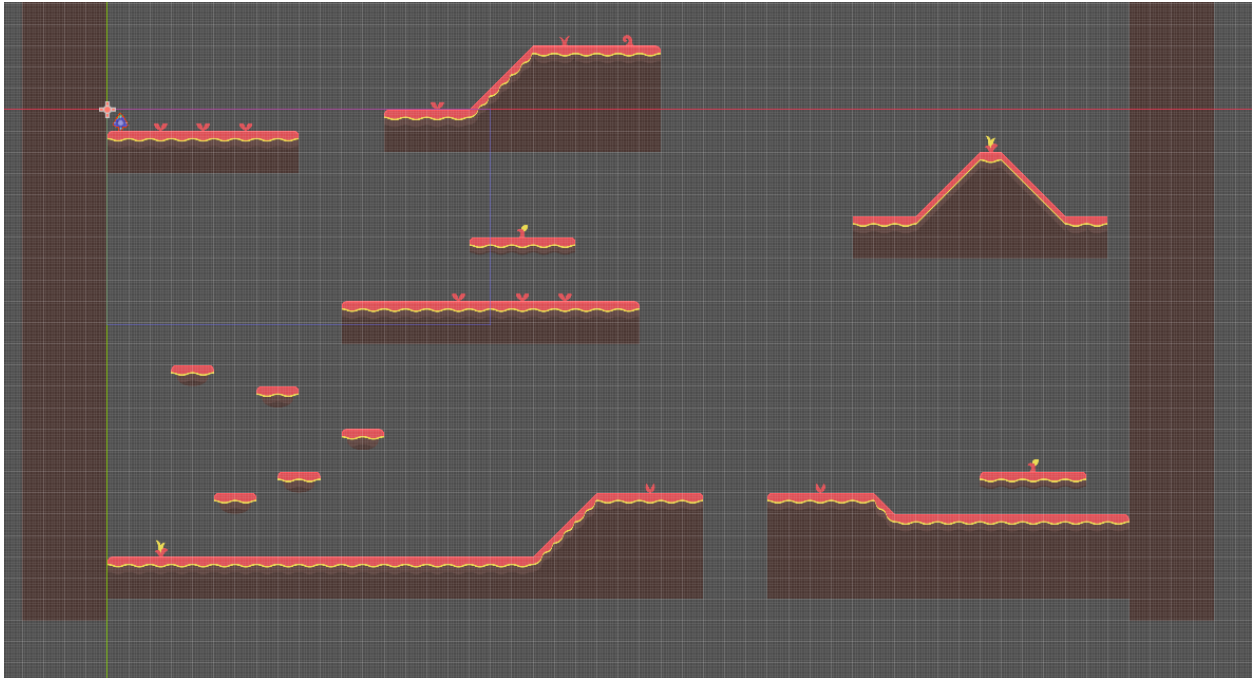
- 
- Update the collision layer appropriately, we will rest the tiles on the 'World' collision layer



- 
- Next, back at the tileset, you go into 'select' mode while you have your tileset selected, then to your physics tab, physics layer 0, and then select the tiles you want to add a collision shape to.
- You add the collision shape by using the green icon and selecting the 4 points of the square shape to make a collision shape (square in the case of this particular tile)

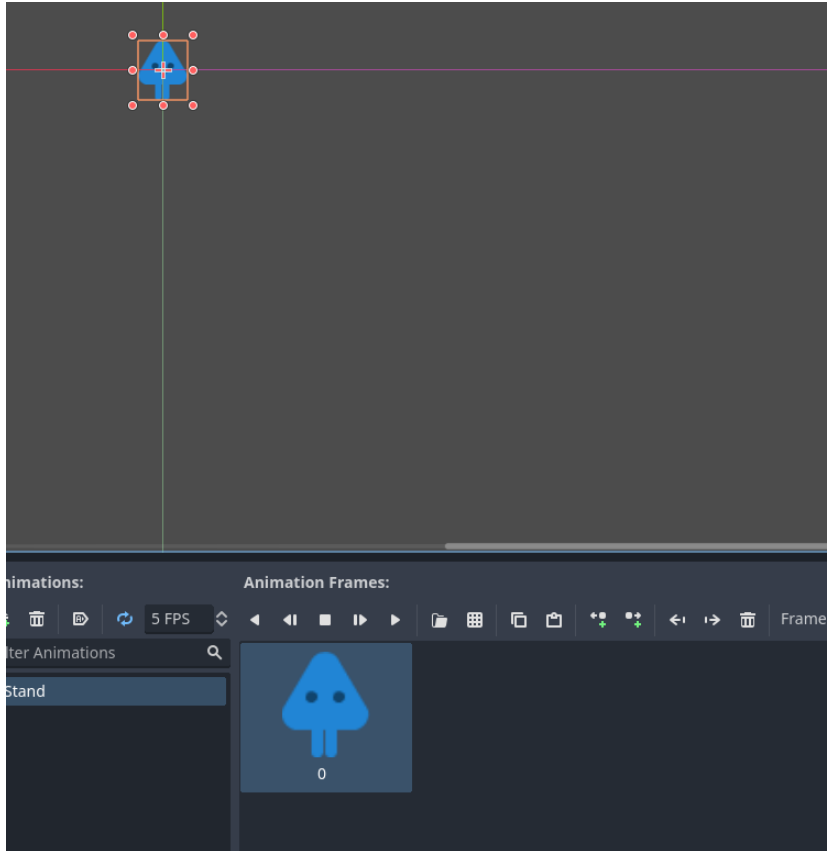


- 
- Now with collisions added to our tiles, our player will properly collide with the tiles and not fall through
- Drag the tilemap scene into the World scene, under the Node2D 'World' node.
- You can now select the tiles you want and start building out your level!

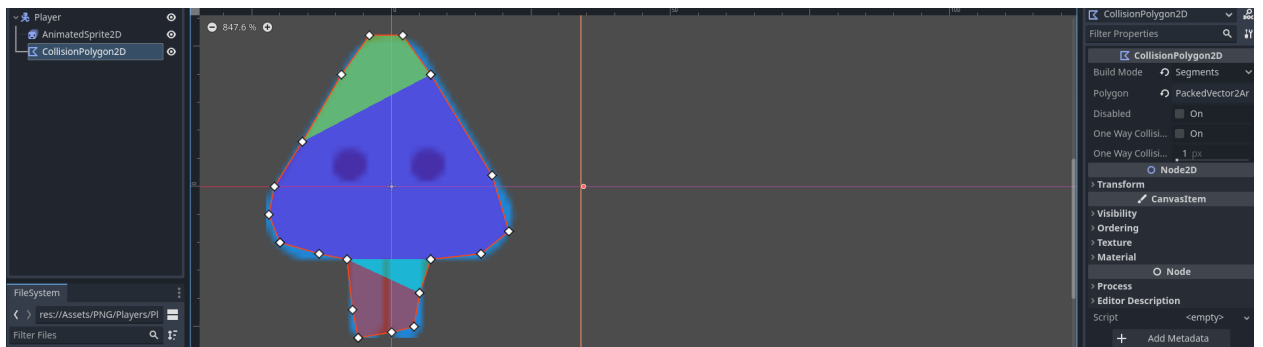
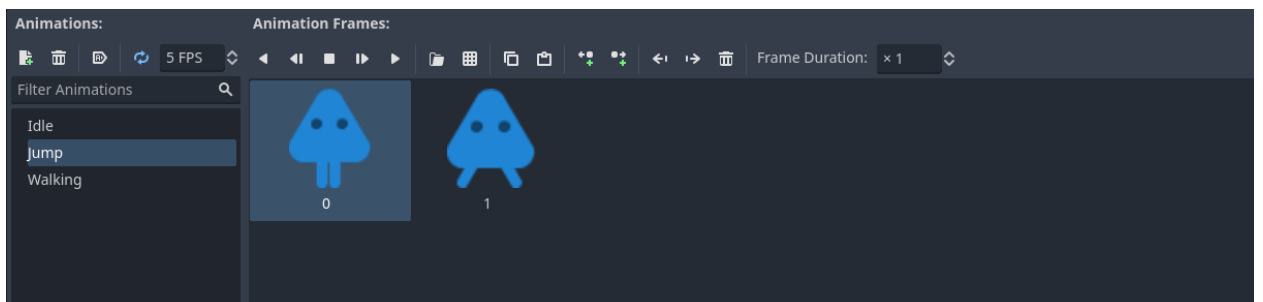


### Sprites - Player & Scripts

- CharacterBody2D node in new scene
- AnimatedSprite2D
- Open the image *PNG\Players\Player Blue\playerBlue\_stand.png*



- 
- You can add many different Animations and frames



- 
- Use collisionPolygon2D and segments
- Save scene as player scene
- Then add a new script to the Root node

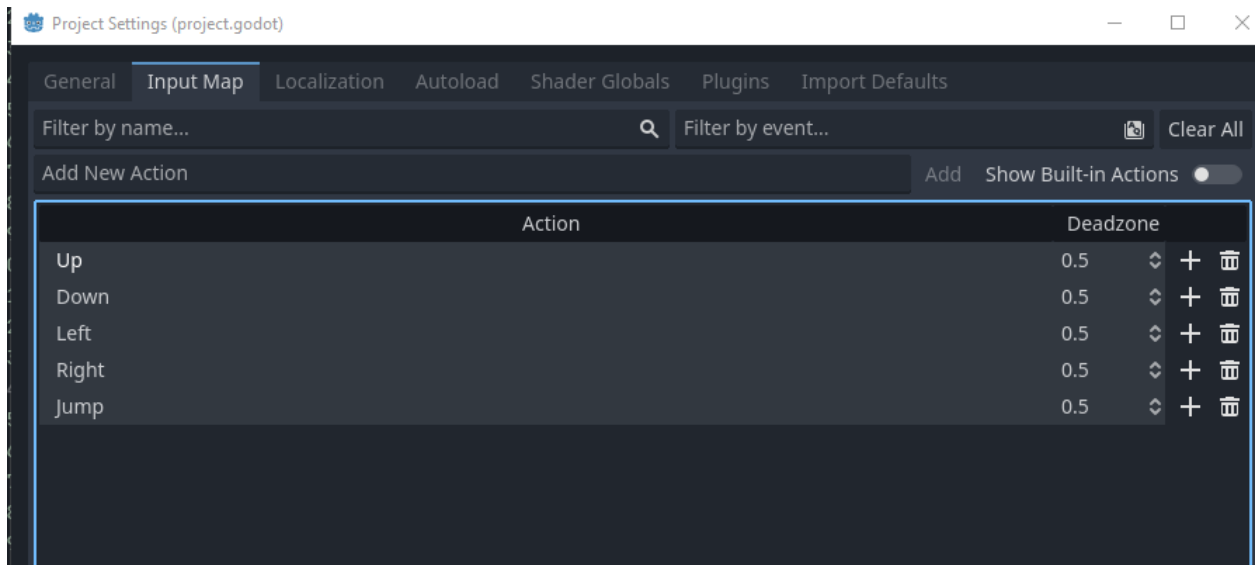
- Name the script 'player'

```

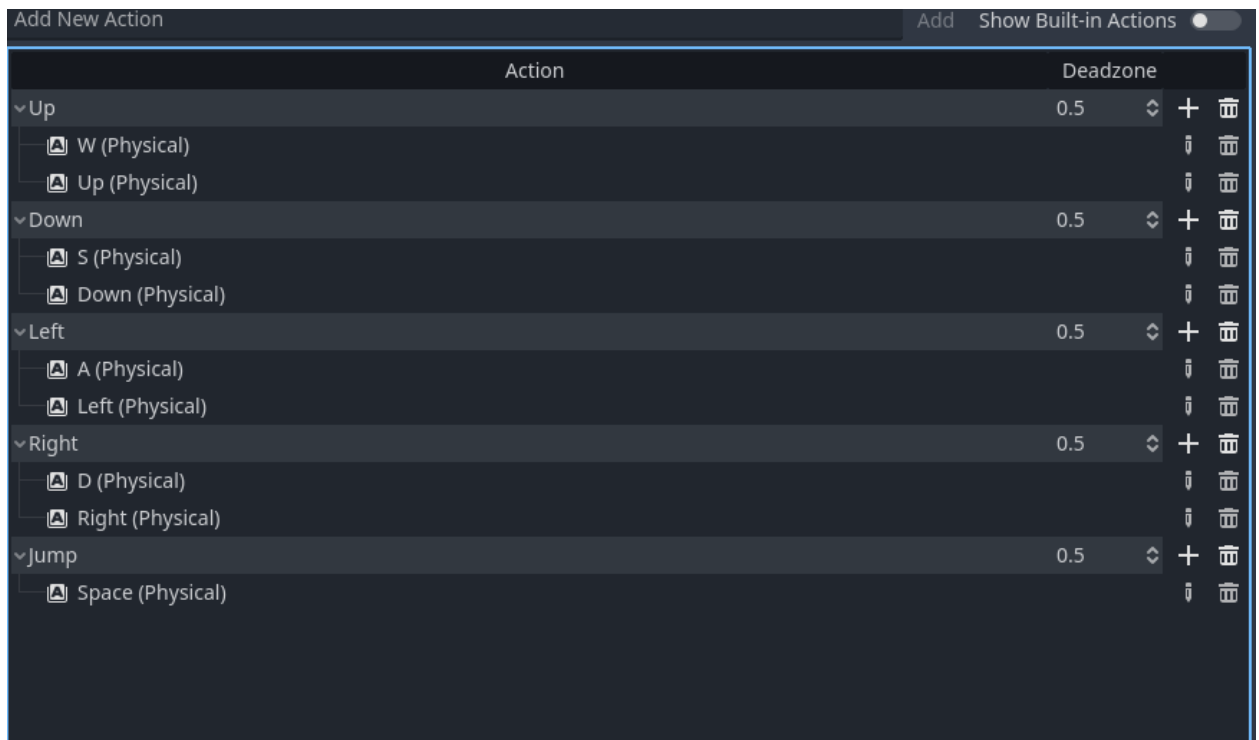
1  extends CharacterBody2D
2
3
4  const SPEED = 300.0
5  const JUMP_VELOCITY = -400.0
6
7  # Get the gravity from the project settings to be synced with RigidBody nodes.
8  var gravity = ProjectSettings.get_setting("physics/2d/default_gravity")
9
10
11 func _physics_process(delta):
12     # Add the gravity.
13     if not is_on_floor():
14         velocity.y += gravity * delta
15
16     # Handle Jump.
17     if Input.is_action_just_pressed("ui_accept") and is_on_floor():
18         velocity.y = JUMP_VELOCITY
19
20     # Get the input direction and handle the movement/deceleration.
21     # As good practice, you should replace UI actions with custom gameplay actions.
22     var direction = Input.get_axis("ui_left", "ui_right")
23     if direction:
24         velocity.x = direction * SPEED
25     else:
26         velocity.x = move_toward(velocity.x, 0, SPEED)
27
28     move_and_slide()
29

```

- 
- The default movement script is mostly okay in terms of simplicity.
- We will change it slightly to use our own input map.
- Navigate to Project Settings and then Input Map, add these 5 actions



- Then press the plus button and add inputs, you can filter them by typing the input you want



- 
- It should look something similar to this

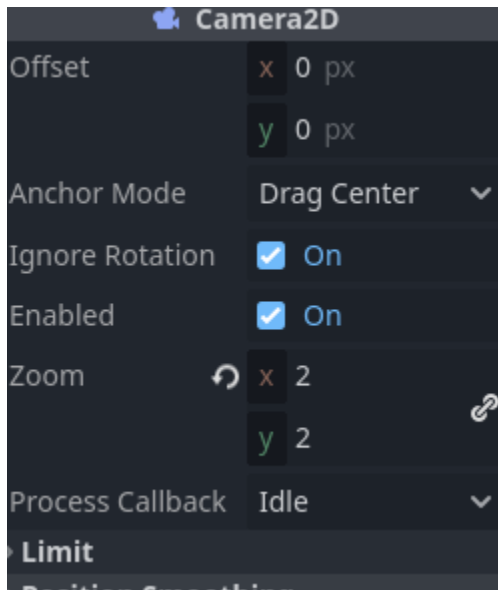
```

3
4  const SPEED = 300.0
5  const JUMP_VELOCITY = -400.0
6
7  # Get the gravity from the project settings to be synced with RigidBody nodes.
8  var gravity = ProjectSettings.get_setting("physics/2d/default_gravity")
9
10
11 func _physics_process(delta):
12     # Add the gravity.
13     if not is_on_floor():
14         velocity.y += gravity * delta
15
16     # Handle Jump.
17     if Input.is_action_pressed("Jump") and is_on_floor():
18         velocity.y = JUMP_VELOCITY
19
20     # Get the input direction and handle the movement/deceleration.
21     # As good practice, you should replace UI actions with custom gameplay actions.
22     var direction = Input.get_axis("Left", "Right")
23     if direction:
24         velocity.x = direction * SPEED
25     else:
26         velocity.x = move_toward(velocity.x, 0, SPEED)
27
28     move_and_slide()
29

```

- 
- Change input to look like this in script

- Our character has basic movement!
- Add a camera2D to the player in the player scene
- Change the zoom to 2, 2 (So that it's not too zoomed out) - Can adjust as needed



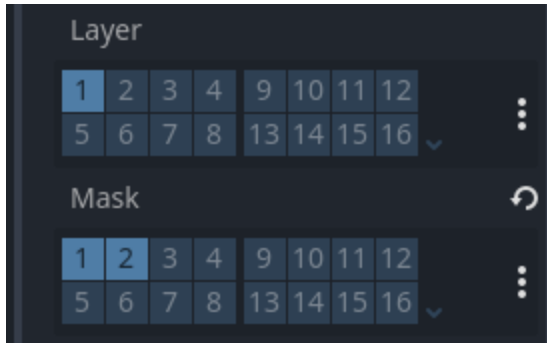
- 
- Back at the script, we can make use of our animated frames in code.

```

2
3 func _physics_process(delta):
4     # Add the gravity.
5     if not is_on_floor():
6         velocity.y += gravity * delta
7
8     # Handle Jump.
9     if Input.is_action_pressed("Jump") and is_on_floor():
10        velocity.y = JUMP_VELOCITY
11
12    if Input.is_action_pressed("Right"):
13        $AnimatedSprite2D.play("Walking")
14        $AnimatedSprite2D.flip_h = false
15    elif Input.is_action_pressed("Left"):
16        $AnimatedSprite2D.flip_h = true
17        $AnimatedSprite2D.play("Walking")
18    else:
19        $AnimatedSprite2D.play("Idle")
20
21    #if not is_on_floor():
22        $AnimatedSprite2D.play("Jump")

```

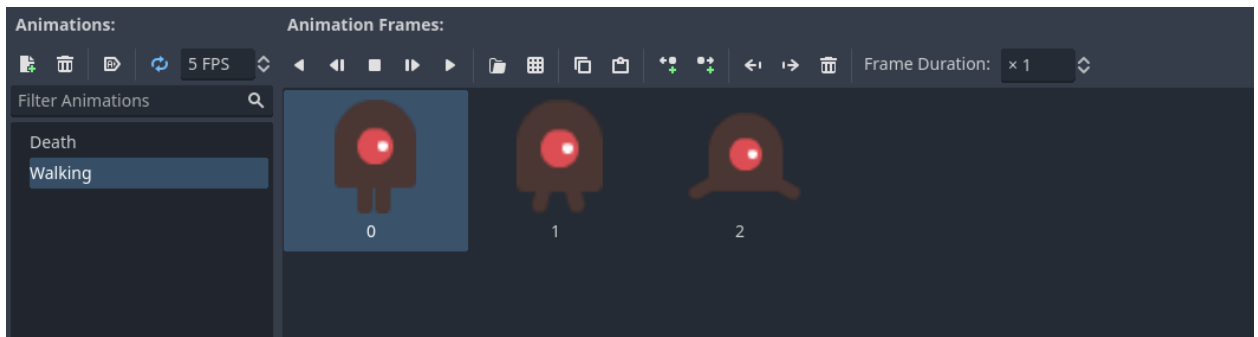
- 
- Lastly, update the collision layer and mask



- 
- The mask layer labeled '2' is the world mask, where the platforms from our tileset rest on, this will allow the player to check for collisions with that layer, and make sure it doesn't fall through.
- Go to the world scene and bring in your player scene to test out the input.

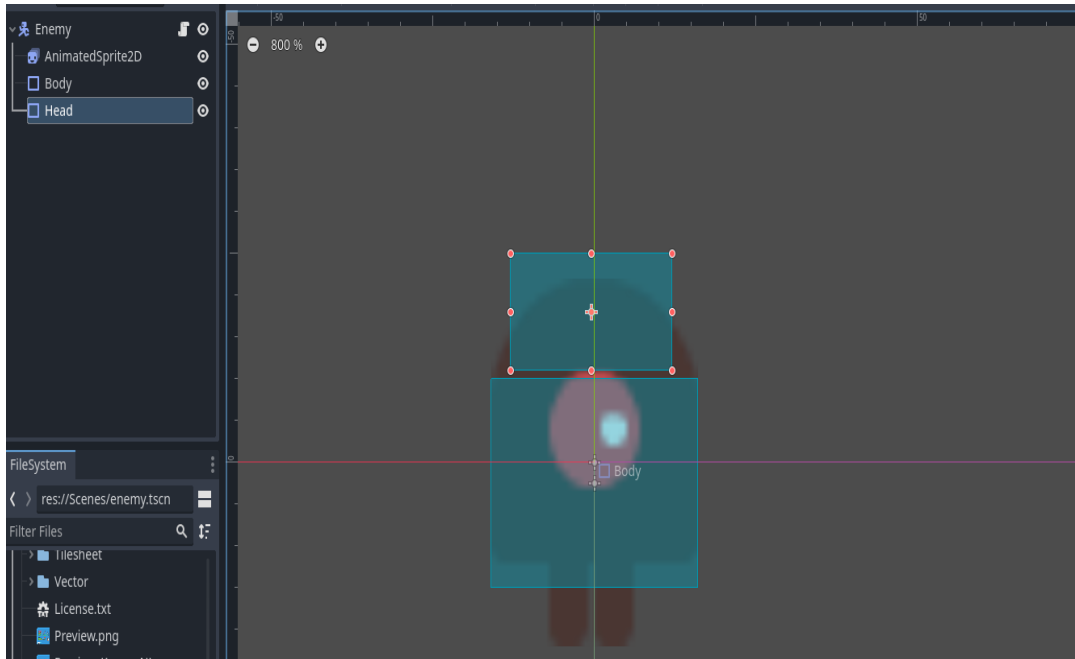
## Enemy:

- Create a new scene, create a CharacterBody2D, add an AnimatedSprite2D, and two CollisionShape2D
- Create a new animated sheet, rename the animation to 'Walking' and add the frames.

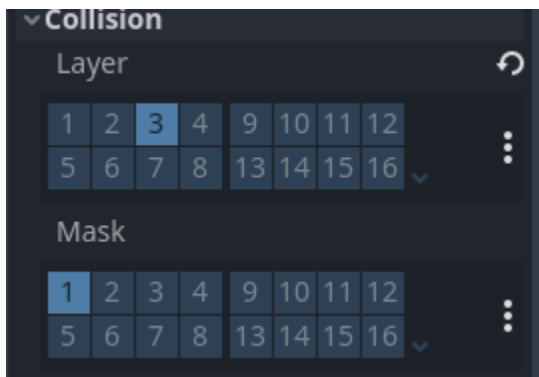


- 
- Create the collision shape, one for the body, and one for the head. The head will be under another Area2D node we will call 'head check'

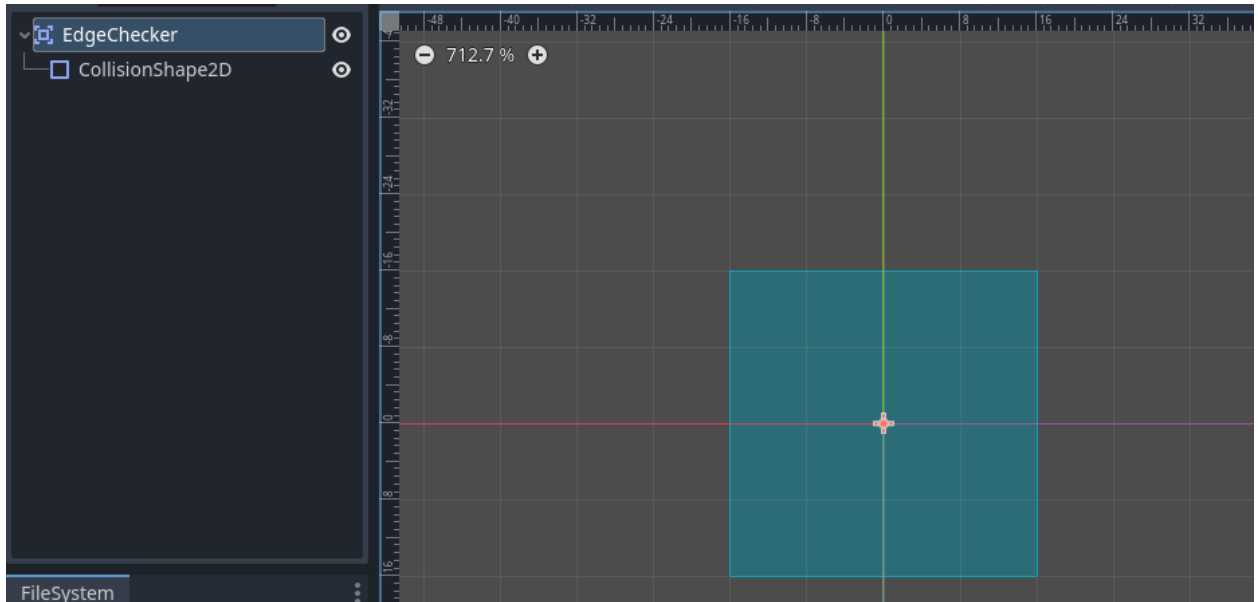




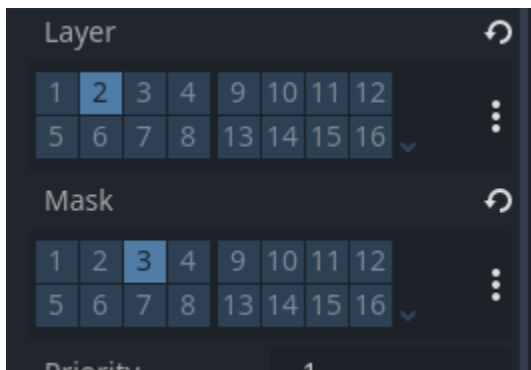
- Update its collision layer and mask, and do the same for the 'head check' as well



- Create a new scene, add an Area2D node, create a CollisionShape2D, make a box. This will act as our 'edge checker' for the enemy. When it hits this edge, we will make the enemy turn around.



- Update the collision layer and mask



- Go back to the enemy scene and attach a script
- In the script we will add basic behavior

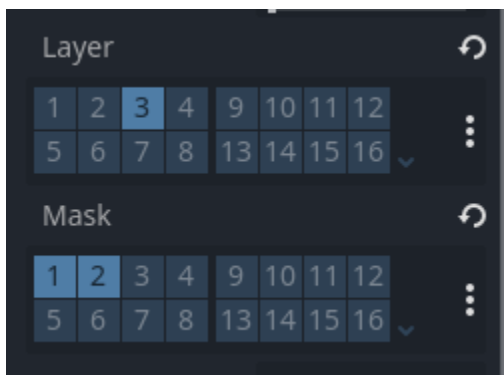
```
1  extends CharacterBody2D
2
3  @export var direction = 1
4
5  ▾ func _ready():
6    >| #Checks if the sprite should be flipped or not at the start
7  ▾ >| if direction == 1:
8    >| >| $AnimatedSprite2D.flip_h = true
9  ▾ >| else:
10   >| >| $AnimatedSprite2D.flip_h = false
11   >|
12
13 ▾ func _physics_process(delta):
14   >| velocity.x += 20
15   >|
16   >| #Could write it simply like this
17 ▾ >| if direction == 1:
18   >| >| velocity.x = -40
19 ▾ >| elif direction == -1:
20   >| >| velocity.x = 40
21   >|
22 ▾ >| #You can write it this way as well
23   >| #velocity.x = 40 * direction
```

```

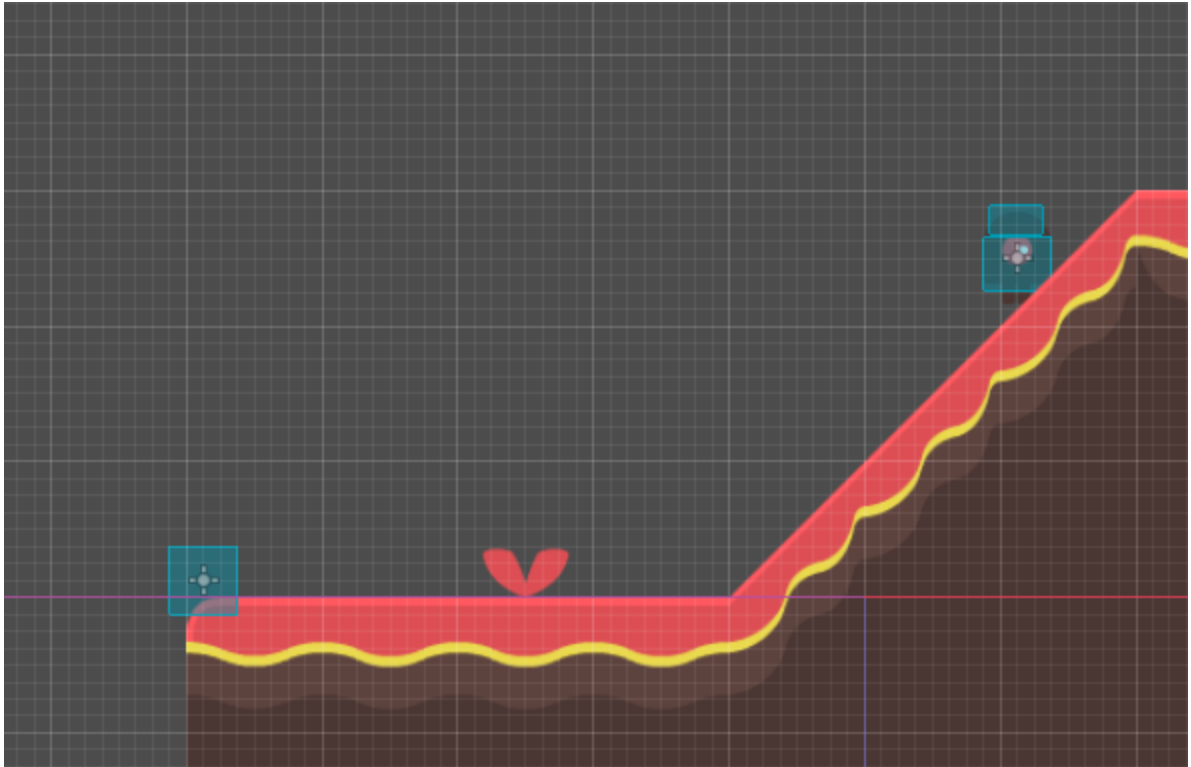
13  ▸ func _physics_process(delta):
14  ▸     velocity.x += 20
15  ▸
16  ▸     #Could write it simply like this
17  ▸     if direction == 1:
18  ▸         velocity.x = -40
19  ▸     elif direction == -1:
20  ▸         velocity.x = 40
21  ▸
22  ▸     #You can write it this way as well
23  ▸     #velocity.x = 40 * direction
24  ▸
25  ▸
26  ▸     move_and_slide()
27

```

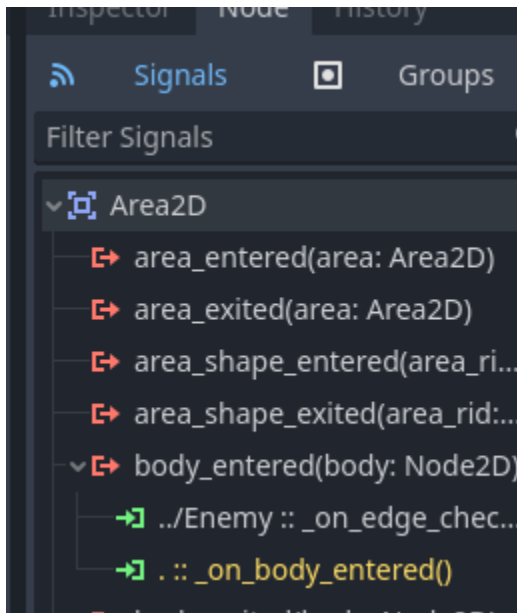
- 
- Update the collision layer and mask for your enemies



- 
- Go to the world scene and add your enemy or enemies.
- Then add an edge checker to the edges of your world.



- 
- In the world scene, click on the edge checker and go to the node tab to bring up signals.
- Click on body entered and connect it to the Enemy script.



- 
- Once connected update the script to turn the enemy around. We can add walking animation as well to the script.

```
1  extends CharacterBody2D
2
3  @export var direction = 1
4  @export var WALK_SPEED = 65
5
6  # Get the gravity from the project settings to be synced with RigidBody nodes.
7  var gravity = ProjectSettings.get_setting("physics/2d/default_gravity")
8
9  func _ready():
10     #Checks if the sprite should be flipped or not at the start
11     if direction == 1:
12         $AnimatedSprite2D.flip_h = true
13     else:
14         $AnimatedSprite2D.flip_h = false
15
16
17  func _physics_process(delta):
18
19     if not is_on_floor():
20         velocity.y += gravity * delta
21
22     velocity.x += 20
23
24     #Could write it simply like this
25     if direction == 1:
26         velocity.x = WALK_SPEED * -1
27         $AnimatedSprite2D.play("Walking")
28     elif direction == -1:
29         velocity.x = WALK_SPEED
30         $AnimatedSprite2D.play("Walking")
31
```

```

24 >| #Could write it simply like this
25 ▾>| if direction == 1:
26 >| >| velocity.x = WALK_SPEED * -1
27 >| >| $AnimatedSprite2D.play("Walking")
28 ▾>| elif direction == -1:
29 >| >| velocity.x = WALK_SPEED
30 >| >| $AnimatedSprite2D.play("Walking")
31 >|
32 ▾>| #You can write it this way as well
33 >| #velocity.x = 40 * direction
34 >|
35 >| #Wall Direction
36 ▾>| if is_on_wall():
37 >| >| direction = direction * -1
38 >|
39 >|
40 >| move_and_slide()
41
42
43
44 ▾> func _on_edge_checker_body_entered(body):
45 ▾>| if body.is_in_group("Enemies") && direction == 1:
46 >| >| direction = -1
47 ▾>| elif body.is_in_group("Enemies") && direction == -1:
48 >| >| direction = 1
49

```

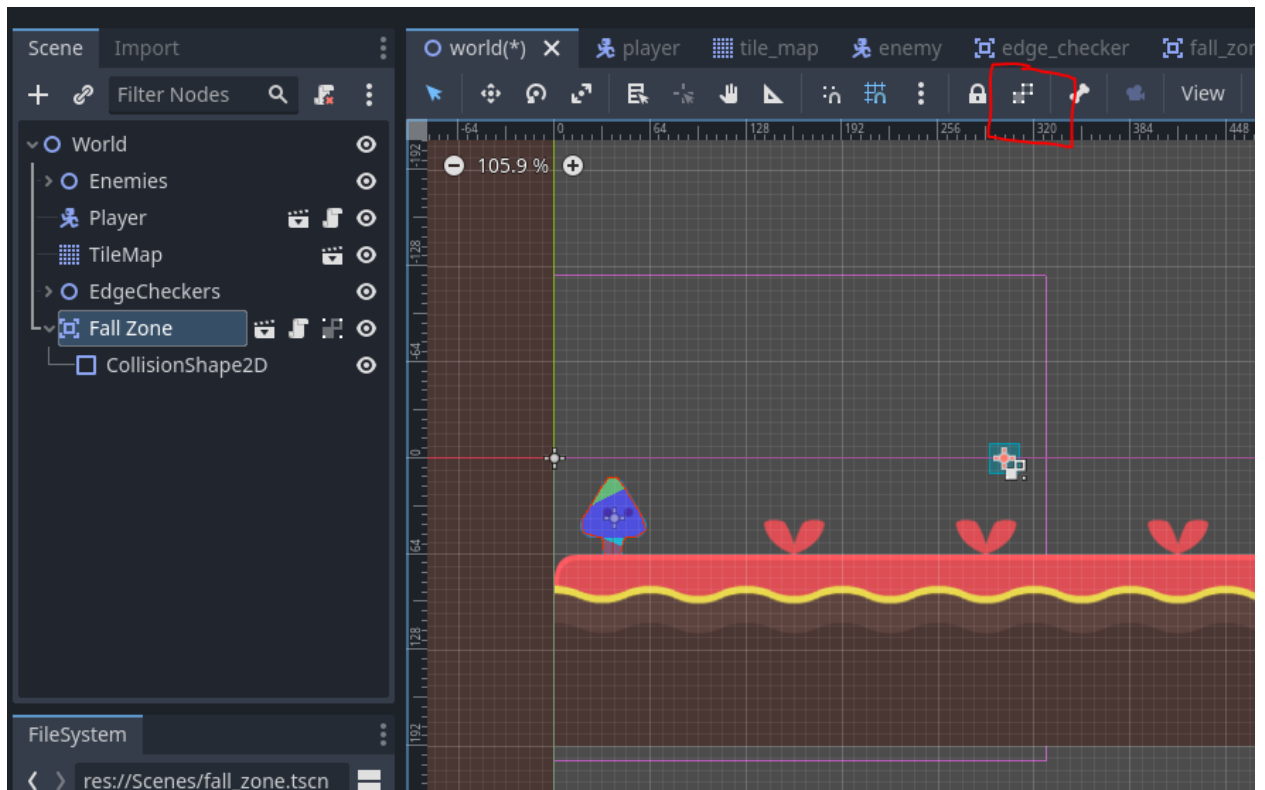
- 
- Edge checkers will now flip the enemy when it reaches an edge

### **Fall Zone:**

- Create a new scene, add an Area2D node, name it “Fall Zone”. We won’t attach a CollisionShape2D in it’s own scene, so it is flexible and reusable in any scene, for example if there’s another level.
- Attach a script to it
- Save the scene
- Update the collision layer so it’s on the ‘World’ Layer, and detects the ‘Player’ mask



- 
- Put the Fall zone in the World scene and make it large enough to ‘catch’ the player by adding a CollisionShape2D

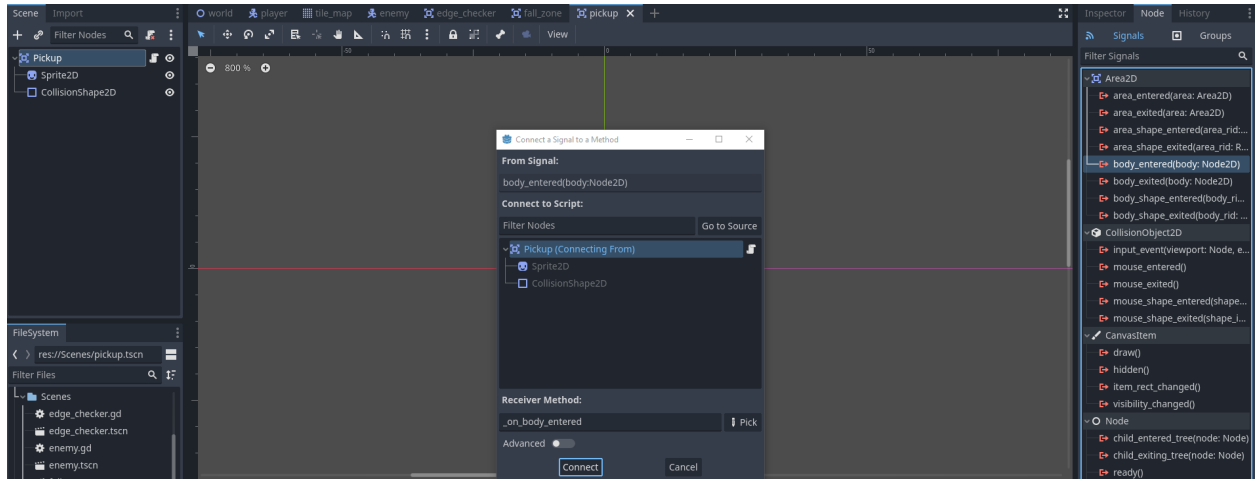


- 
- Once you’ve added it to the World scene, you can click on that icon to make the children not editable, so if you move the fallzone, you won’t accidentally only move the CollisionShape2D





- Attach a script to it
- Add the pickup to the World scene.
- Go back to the original pickup scene
- Click on the root node, then go to the nodes tab, we're going to use signals again.

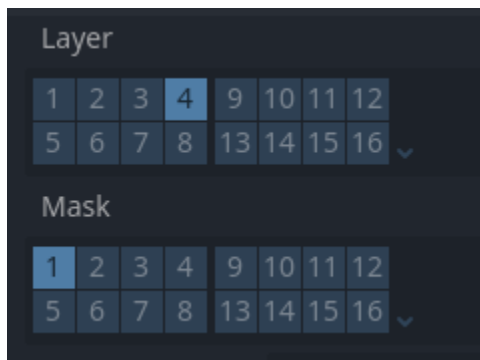


- In the script, you will add just one line for now

```

1  extends Area2D
2
3
4
5  func _on_body_entered(body):
6      # "Deletes" the pickup
7      queue_free()
8  
```

- However, we will have to change the collision layer and mask, because right now if an enemy walks into this pickup, it will also despawn the pickup.
- It will rest on the 'Pickup' layer, but look for the 'Player' in the mask.



- Now we want to track the pickup, when it's actually interacted with by the player.
- Go back to the player script, and add a new variable at the top.

```

1  extends CharacterBody2D
2
3  var pickups = 0
4
5  const SPEED = 300.0
6  const JUMP_VELOCITY = -500.0
7
8  @onready var animatedSprite = $AnimatedSprite
9
10 # Get the gravity from the project settings to b
11 var gravity = ProjectSettings.get_setting("physi

```

- Then add a new function at the bottom of the script

```

45 func add_pickup():
46     pickups = pickups + 1
47

```

- Go to the pickup's script

```

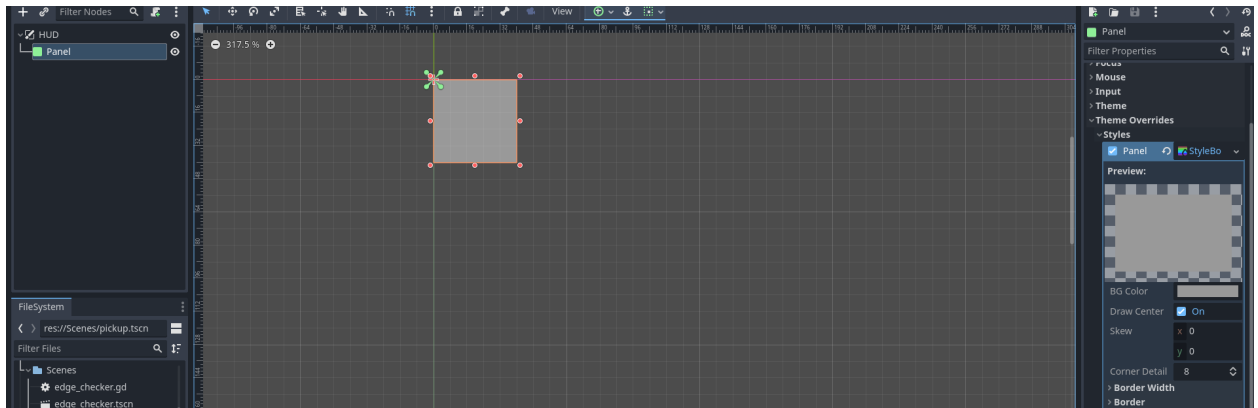
1  extends Area2D
2
3
4
5  func _on_body_entered(body):
6      #Deletes" the pickup
7      queue_free()
8      #Player's code, because he is the body that we are interacting with
9      body.add_pickup()
10

```

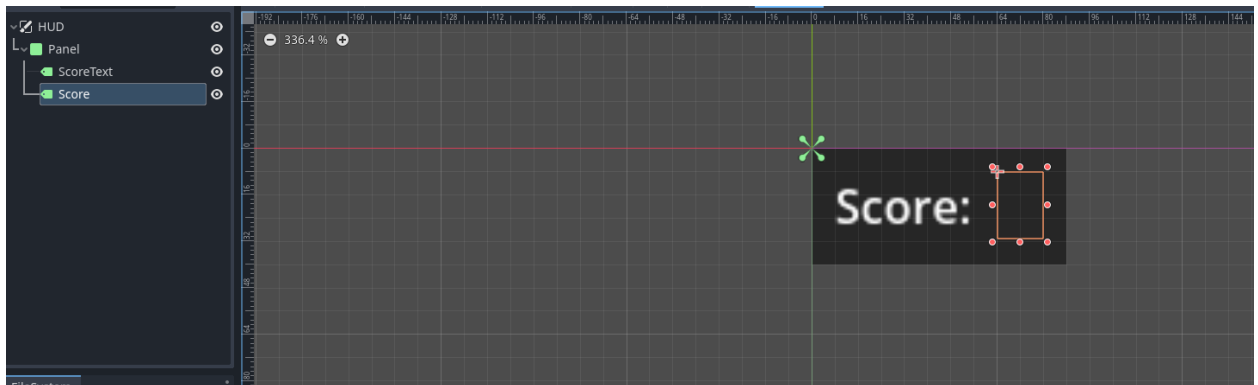
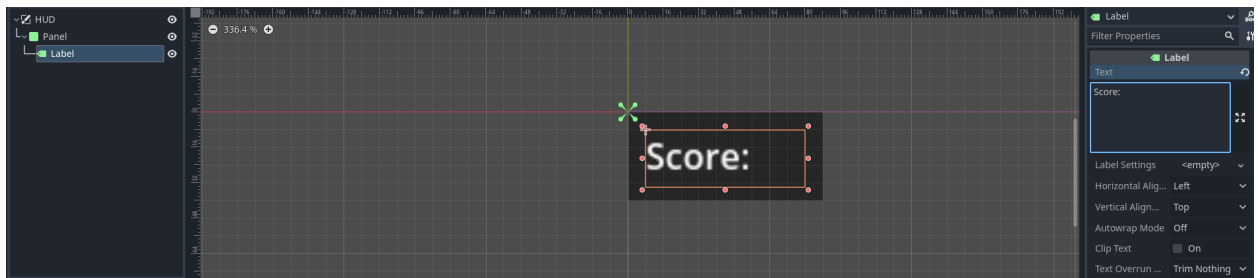
- Add this line
- It will call the function from the player's script, if the player interacts with the pickup.

### UI:

- Next we want to visually see when our score is increased by picking up the pickup.
- Create a new scene using a canvas layer node, name it 'HUD'
- Then add a Panel child node
- Go to the right and while you have the node selected, go under theme, and style
- Next select a StyleboxFlat to edit your panel
- You can add a background a color, change the opacity, etc...



- Next add two labels as a child of panel



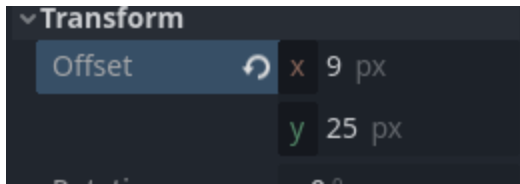
- Add a script to the root node (The “HUD”)

```

1  extends CanvasLayer
2
3  var pickups = 0
4
5  func _ready():
6      $Panel/Score.text = str(pickups)
7

```

- Add the HUD scene to the World scene, transform it so it appears in the viewport appropriately



- Place the pickups in the World Scene, just place one for now. You can use a Node2D to organize your nodes in 'folders'



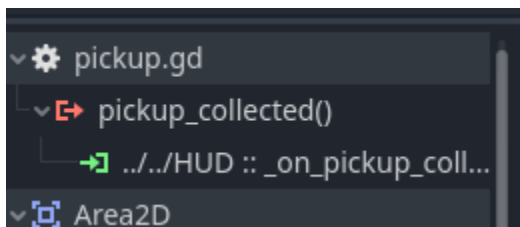
- We will add a custom signal here, go to the script.

```

1  extends Area2D
2
3  signal pickup_collected
4
5  func _on_body_entered(body):
6      >| #Deletes" the pickup
7      >| queue_free()
8      >| emit_signal("pickup_collected")
9      >|
10     >| #Player's code, because he is the body that we are interacting with
11     >| body.add_pickup()
12

```

- 
- Then connect the custom signal to the HUD's script, rename the signal to make it shorter.



```

v [gear] pickup.gd
  - [red arrow] pickup_collected()
    [green arrow] ../..HUD :: _on_pickup_coll...
v [blue square] Area2D

```

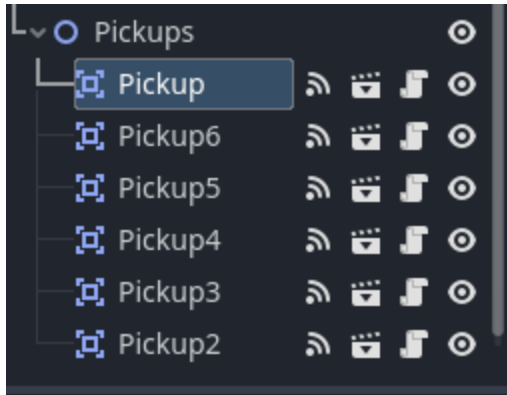
- 
- In the HUD's script, edit the function

```

1  extends CanvasLayer
2
3  var pickups = 0
4
5  func _ready():
6      >| $Panel/Score.text = str(pickups)
7
8
9  func _on_pickup_collected():
10     >| pickups = pickups + 1
11     >| _ready()
12
13

```

- 
- This should update the pickups variable properly when we pick up the pickup now.
- We can now duplicate the pickups in the World scene to create multiple, and it should constantly update.



### Enemy Interaction with Player:

- Go to the Enemy scene, connect a signal for the head check to the enemy script.

```

46
47 func edge_movement(body):
48     if body.is_in_group("Enemies") && direction == 1:
49         direction = -1
50     elif body.is_in_group("Enemies") && direction == -1:
51         direction = 1
52
53
54 func _on_head_check_body_entered(body):
55     $AnimatedSprite2D.play("Death")
56     WALK_SPEED = 0
57

```

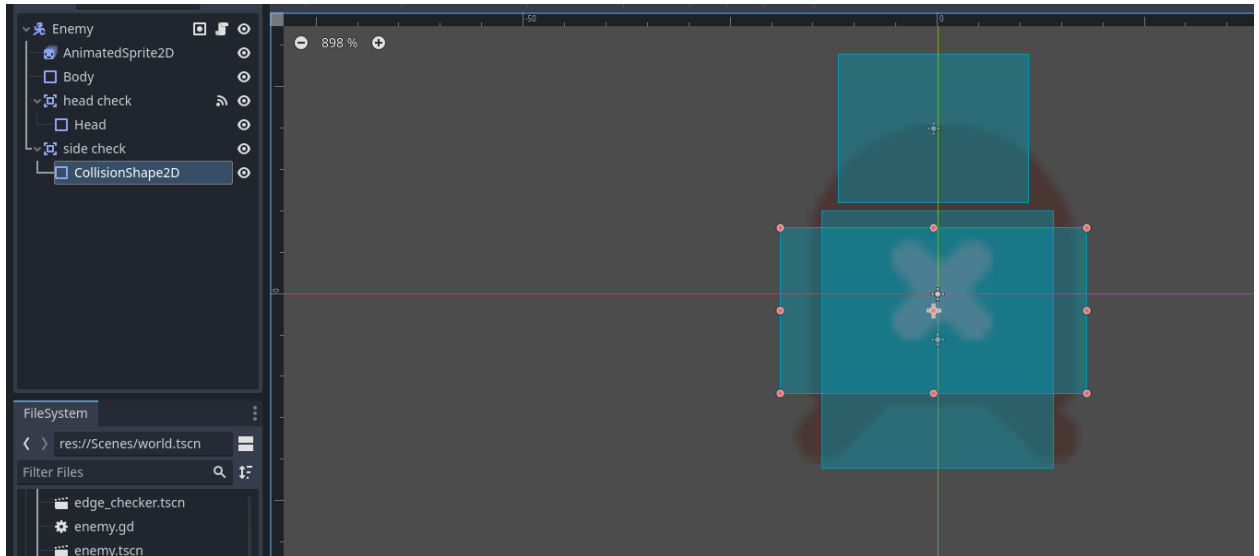
- Now we will make it not collide with the player

```

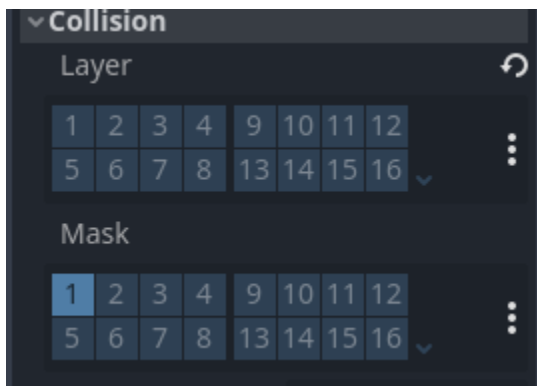
53
54 func _on_head_check_body_entered(body):
55     $AnimatedSprite2D.play("Death")
56     WALK_SPEED = 0
57     #Makes it not collide with the player
58     set_collision_layer_value(3, false)
59     set_collision_mask_value(1, false)
60     $"head check".set_collision_layer_value(3, false)
61     $"head check".set_collision_mask_value(1, false)
62

```

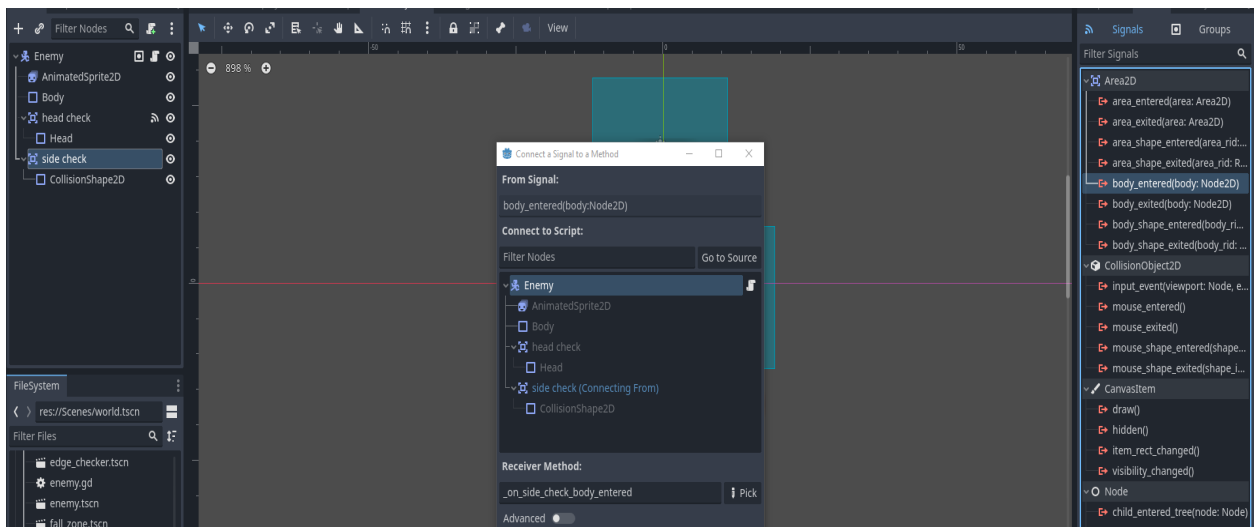
- Go to the enemy scene, and we will edit it
- Create another Area2D and CollisionShape called 'side check' we will later use this to detect player collisions to reload the scene.



- 
- Edit its collision layers and mask appropriately



- 
- Connect a signal



- 
- Update the script

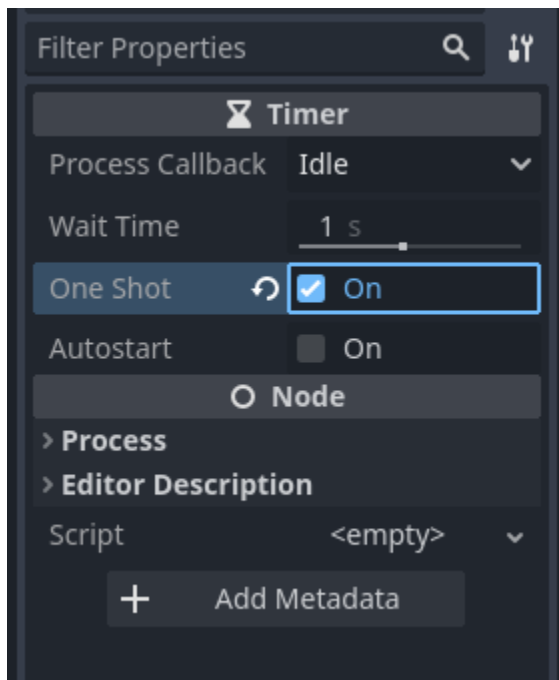


```

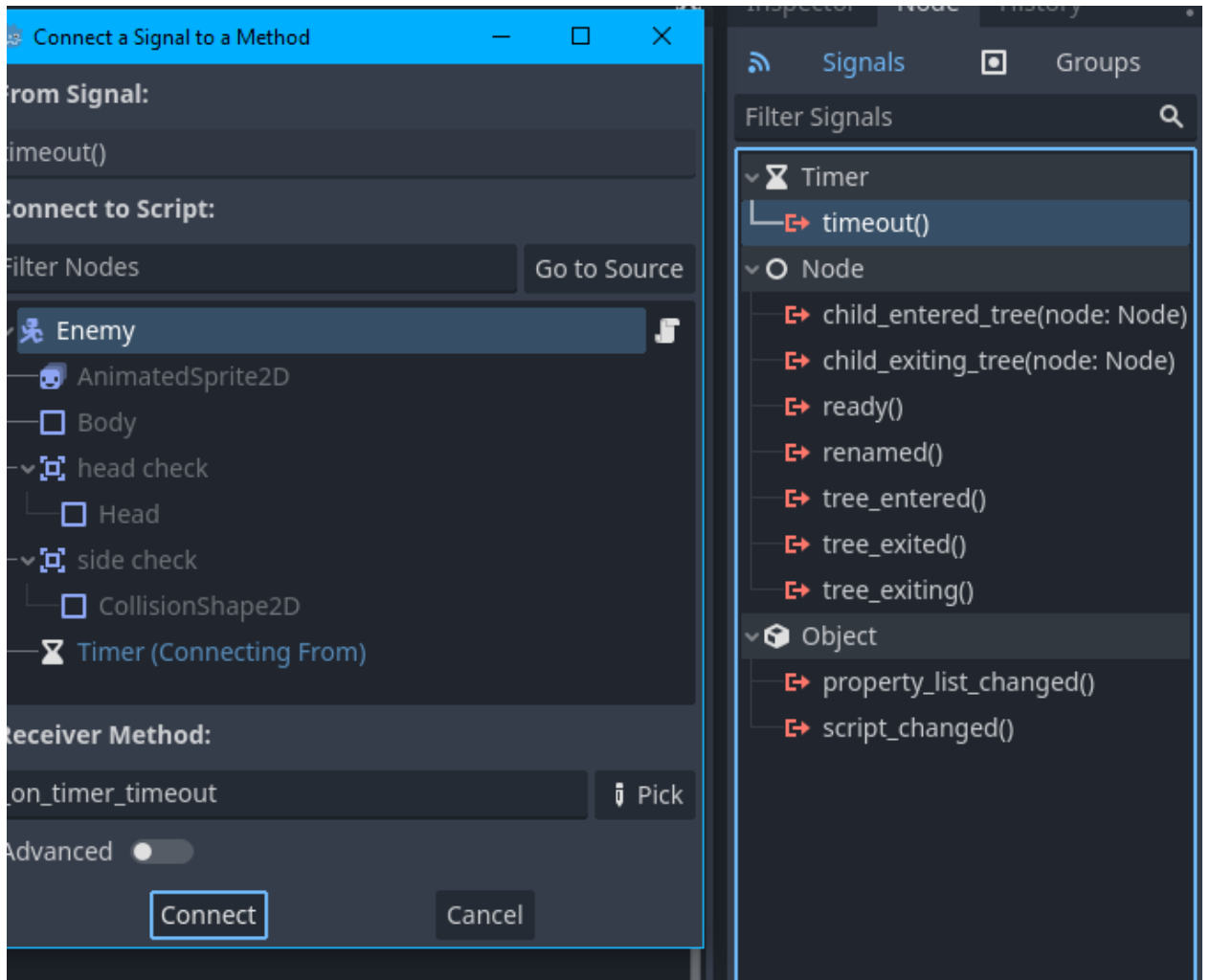
53
54 func _on_head_check_body_entered(body):
55     $AnimatedSprite2D.play("Death")
56     WALK_SPEED = 0
57     #Makes it not collide with the player
58     set_collision_layer_value(3, false)
59     set_collision_mask_value(1, false)
60     $"head check".set_collision_layer_value(3, false)
61     $"head check".set_collision_mask_value(1, false)
62     $"side check".set_collision_layer_value(3, false)
63     $"side check".set_collision_mask_value(1, false)
64
65
66
67
68 func _on_side_check_body_entered(body):
69     get_tree().change_scene_to_file("res://Scenes/world.tscn")
70

```

- 
- Now we will add a timer, and then after a second or so, we will make the enemy disappear when we jump on it
- After you add a timer to the root node of the enemy scene
- Make sure you change it to 'oneshot' so it doesn't restart the timer after the time goes down



- 
-



- 
- Edit the script now

```

54 func _on_head_check_body_entered(body):
55     >I $AnimatedSprite2D.play("Death")
56     >I WALK_SPEED = 0
57     >I #Makes it not collide with the player
58     >I set_collision_layer_value(3, false)
59     >I set_collision_mask_value(1, false)
60     >I $"head check".set_collision_layer_value(3, false)
61     >I $"head check".set_collision_mask_value(1, false)
62     >I $"side check".set_collision_layer_value(3, false)
63     >I $"side check".set_collision_mask_value(1, false)
64     >I $Timer.start()
65     >I
66     >I
67
68
69 func _on_side_check_body_entered(body):
70     >I get_tree().change_scene_to_file("res://Scenes/world.tscn")
71
72
73 func _on_timer_timeout():
74     >I queue_free()
75

```

- 
- With that, we have a working platformer!
- You can add more enemies and edge checkers as needed, though I advise even trying raycasting on the enemy scene instead of an edge checker if you want to make things nicer.