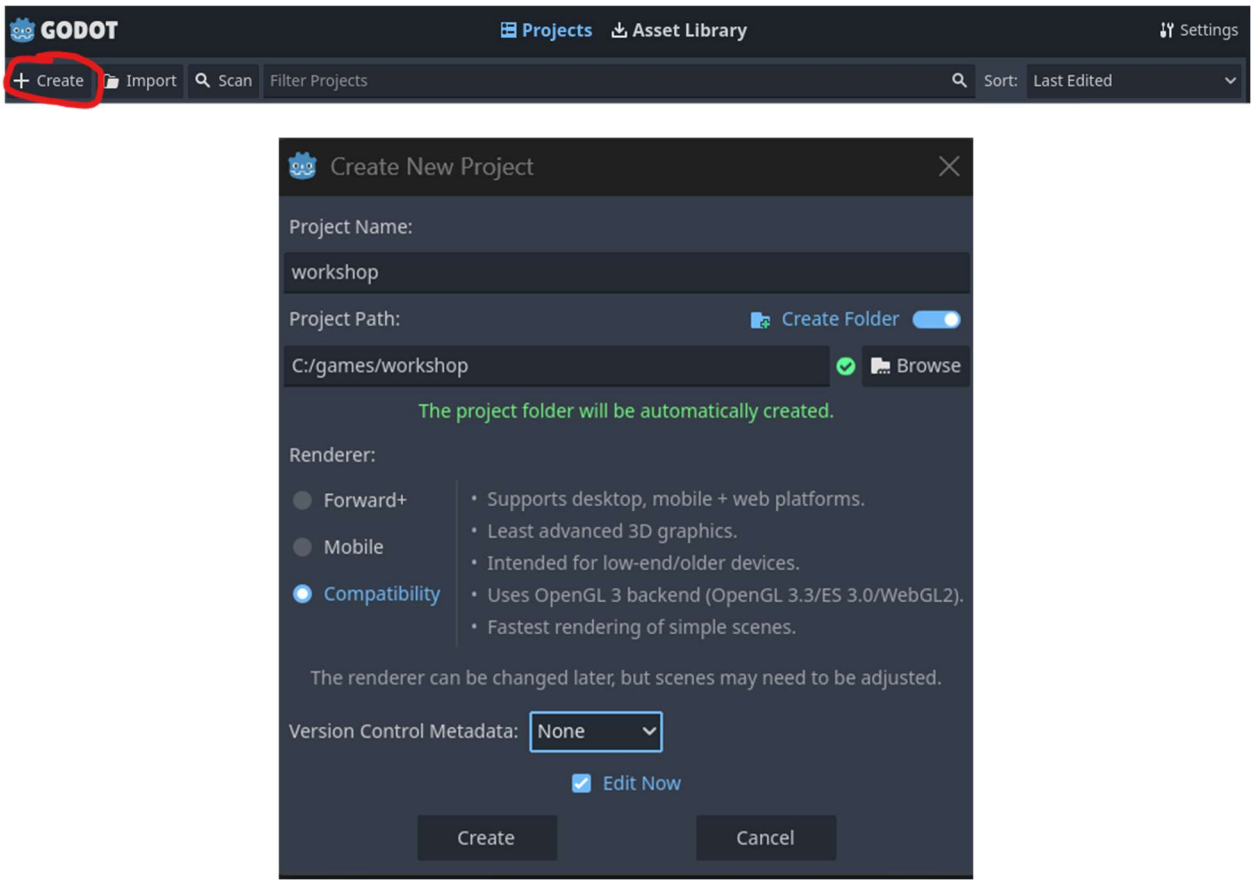
Godot Beginner Tutorial

Creating the Game

Welcome to Godot!

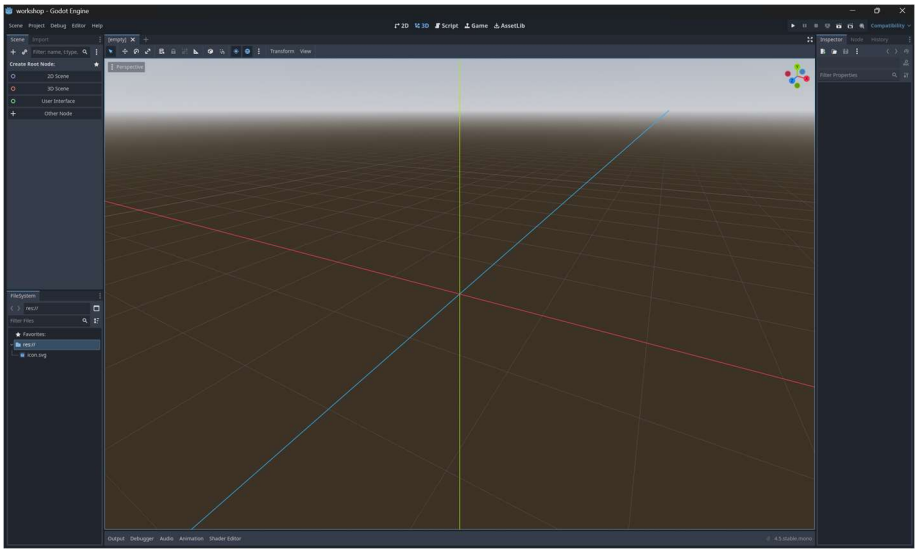
Godot is a wonderful engine that is free to use and completely open source!

When you first launch Godot, you will see the project manager. Here you will want to create a new project.

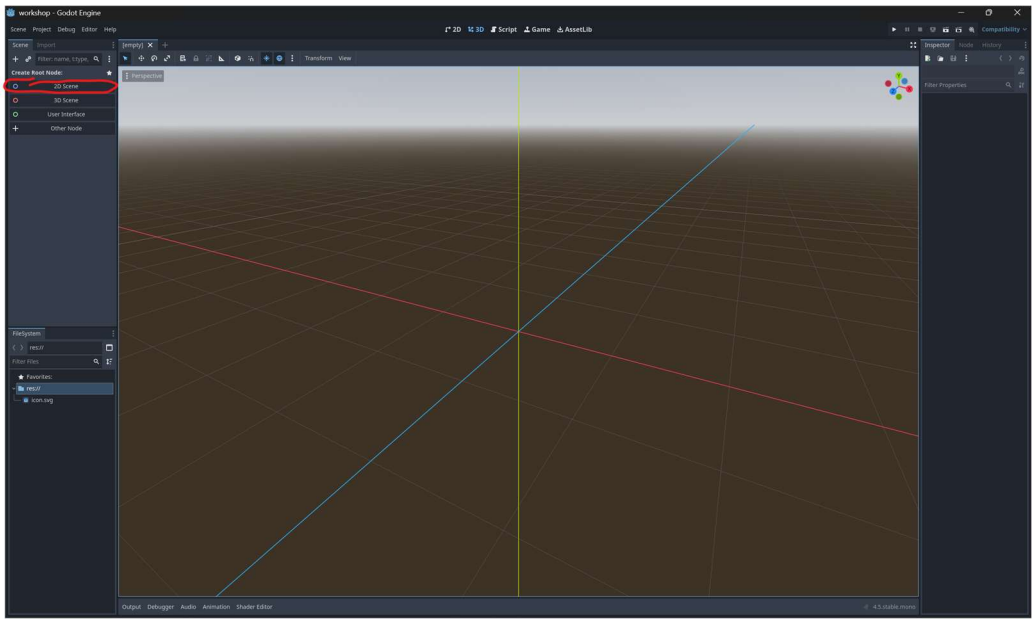


Once you create a new project there will be options for you to customize it.

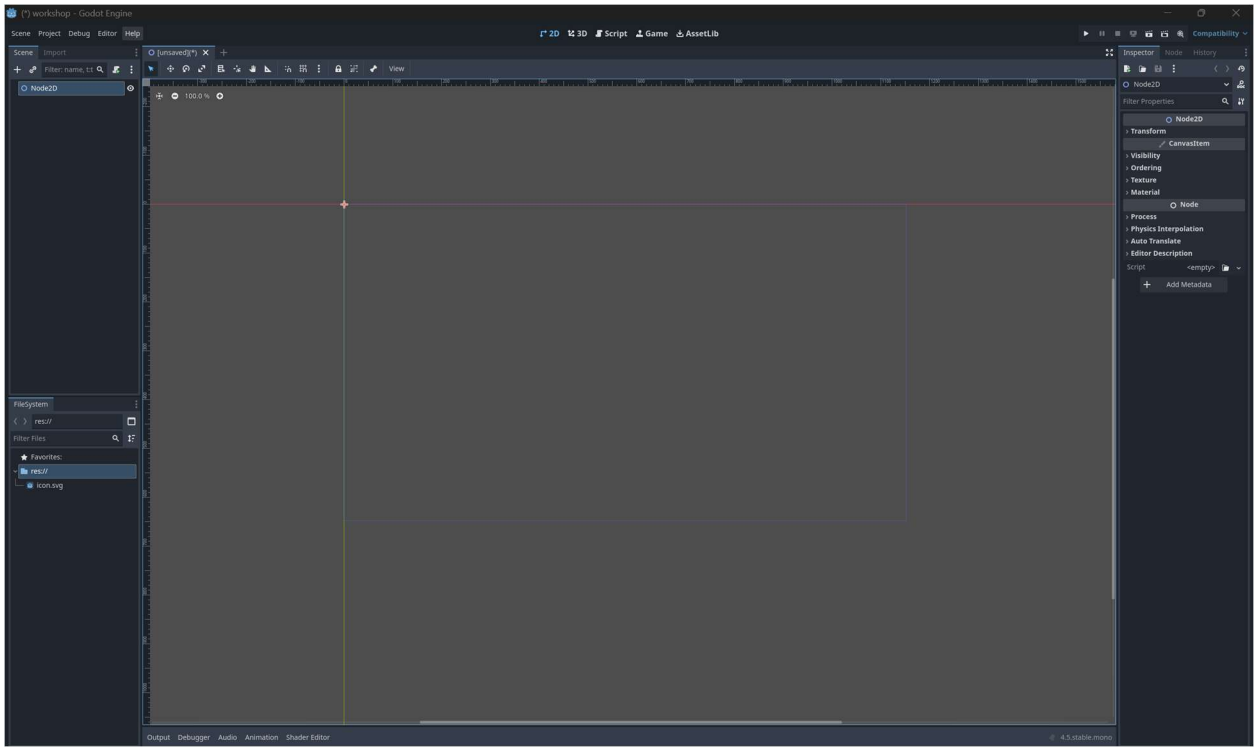
None of the options matter too much for this tutorial so just name it and create it.



Once you are in your project you might feel a little overwhelmed with everything on screen. But not to worry, you will get the hang of it once you start to use it. Right now, we are looking at the 3D editor, but we are making a 2D game so we won't have to worry about this screen again.



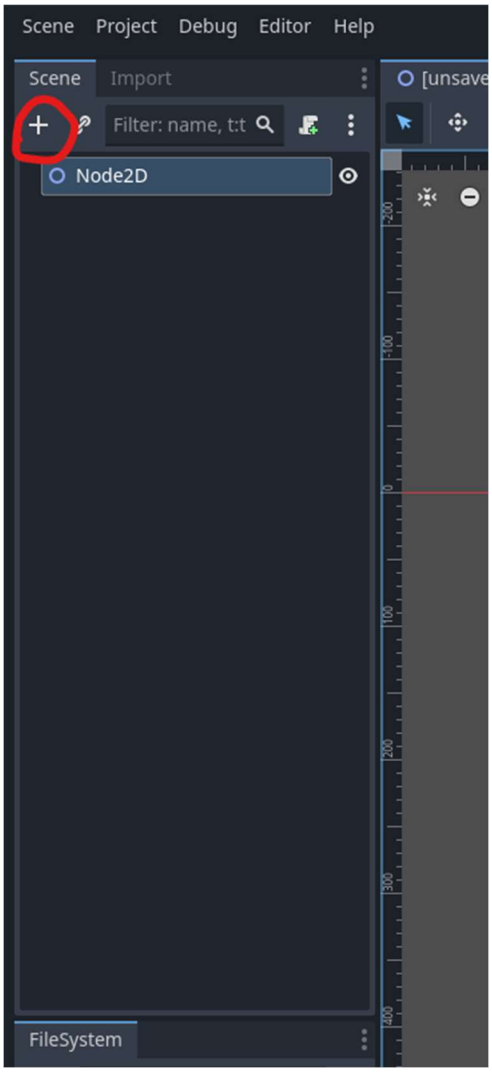
The first thing we will want to do is create a level by creating a new scene. Click on the 2D Node option to create the root of our scene.



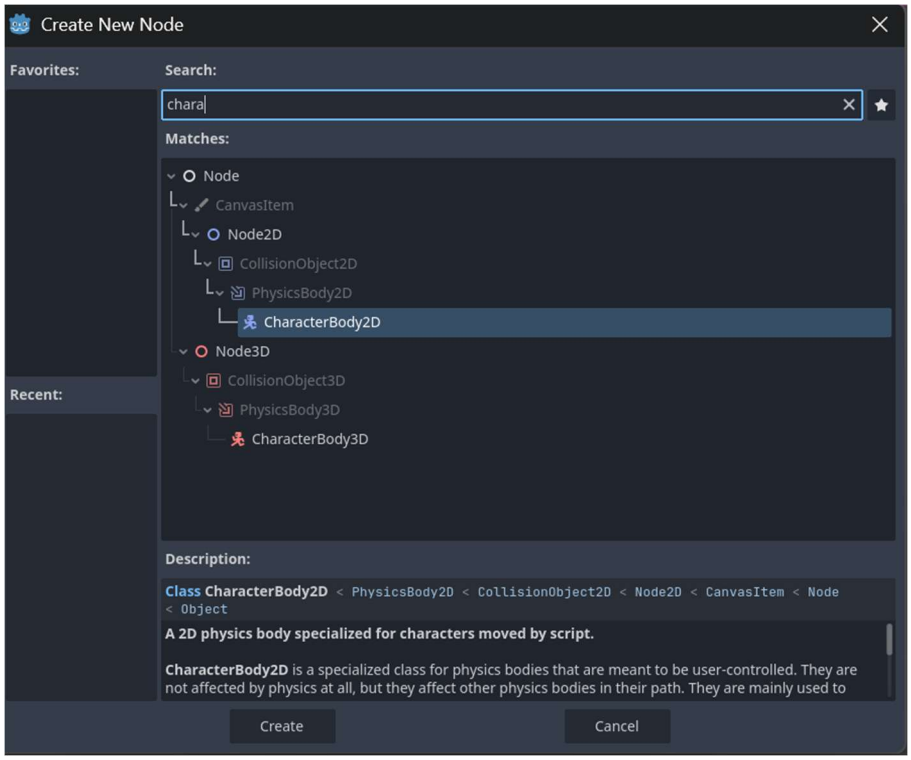
Ah, much better. We are now looking at the 2D editor. On the top left we will also see our “scene tree”. Right now it’s just the 2D node, this will serve as the base for our level to be built on.

It is important to know that in Godot, everything is built from nodes. Simple pieces to make more complicated ones.

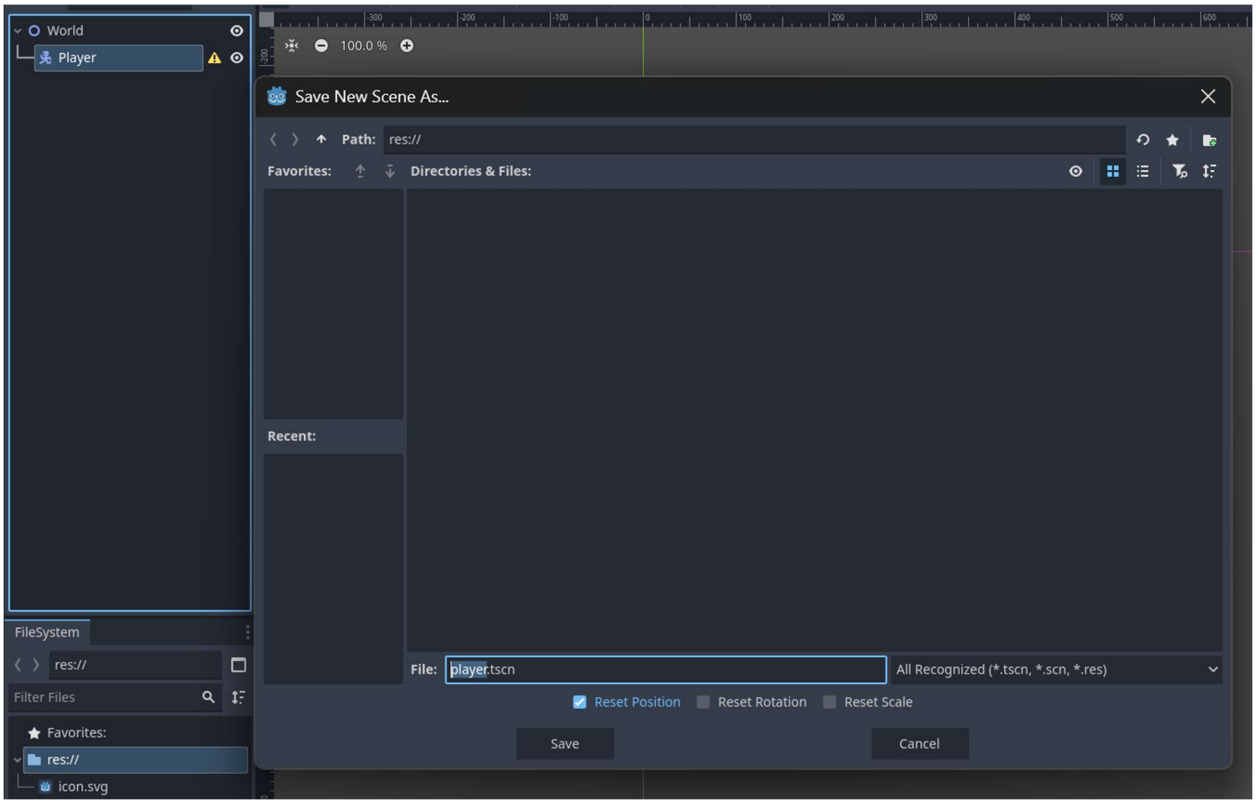
Creating the Player



Let’s make a player for our game. To make a player we will need to add a node to be the child of our “Node2D” root node. To do this click the ‘+’ button on top and search for “CharacterBody2D”

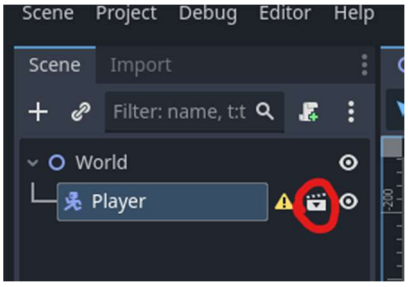


Once we add it, we will see it appear in our scene tree. You can name each node by right clicking on them and clicking the “Rename” button.

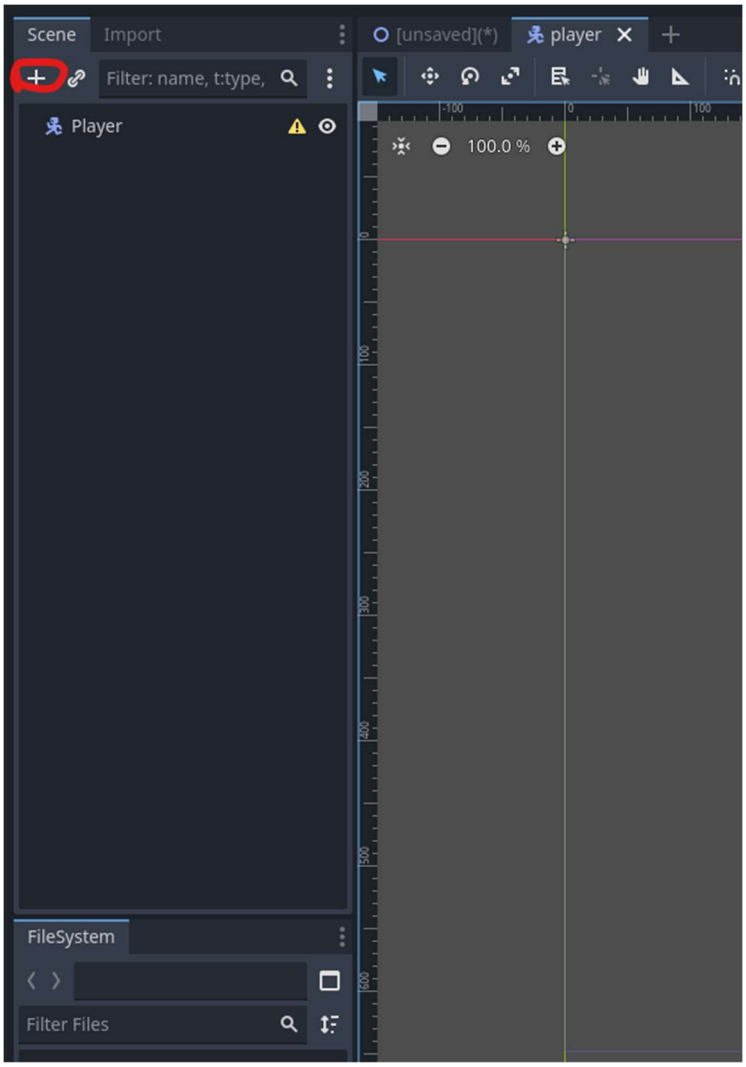


In case we want to make more levels, we should make the player its own scene. Right click on the “Player” node and click “Save Branch as Scene” and save it. That way we can add the player to any new level in the future easier.

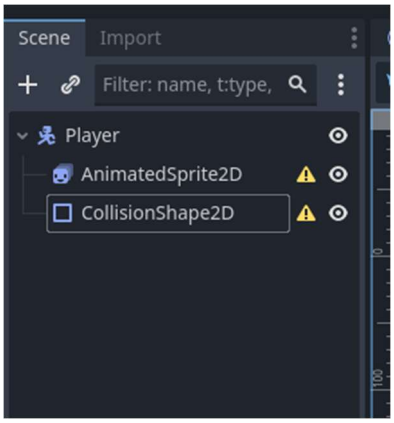
We will now want to edit the player scene, click on the box circled below:



Once again, we are editing a Scene, however this time our “Player” node is the root. We’ll now have to add elements of a player to the scene.



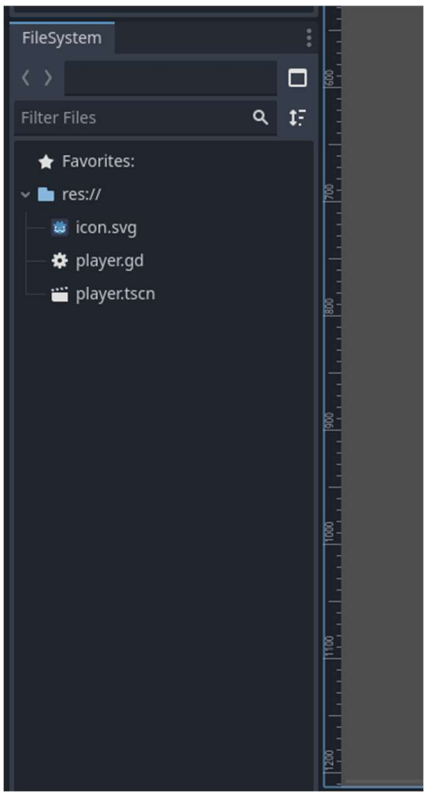
Add a “AnimatedSprite2D” and a “CollisionShape2D” to the Scene.

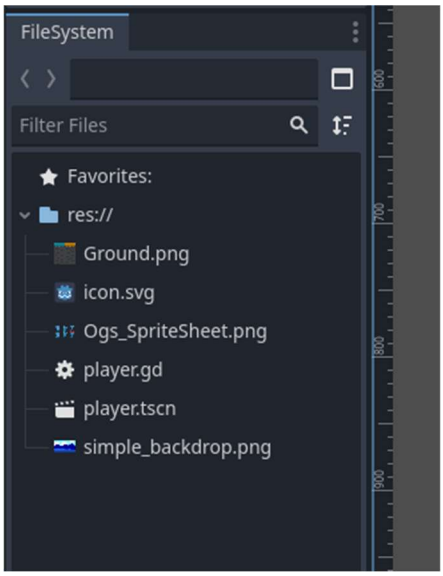


Let’s take a quick look now at our file system. This is where we will find our assets scripts and scenes for our game.

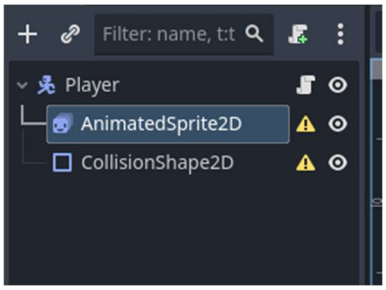
I have these out of order so don’t be alarmed if you don’t see the “player.gd” in your file system yet.

We should add the assets I included into our file system so we may use them in our game. Drag and drop all the pictures into the tab.

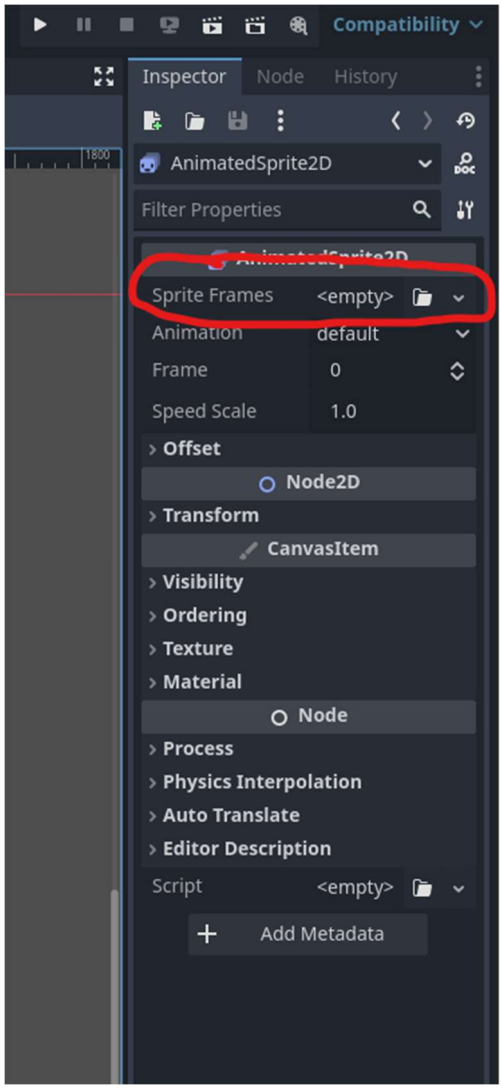


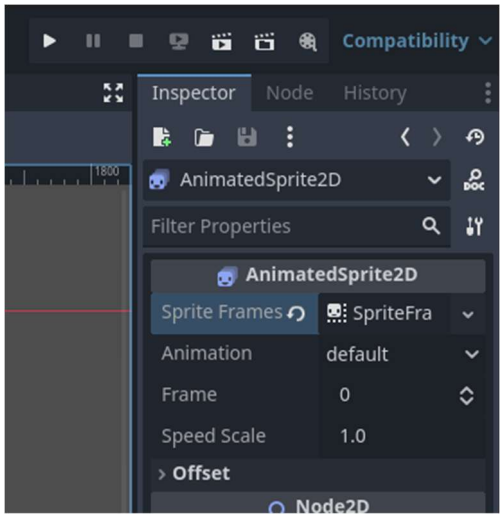


Now let’s add some art so we can see our player. Click on the “AnimatedSprite2D” in the Scene selector.

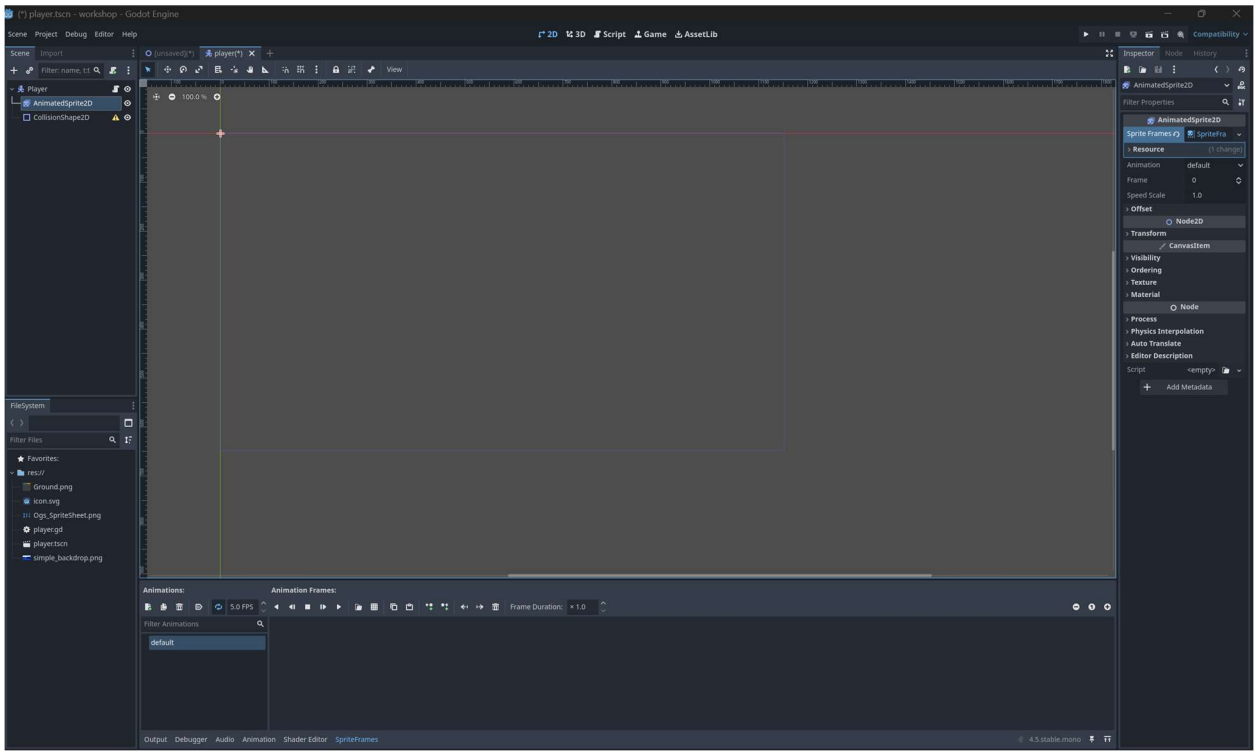


Look now to the right to see the properties of the node you’ve selected. Here we can edit different aspects of each node to suit our game. We will need to add sprite frames, click on the box that says “<empty>” and add a new “SpriteFrames” object.

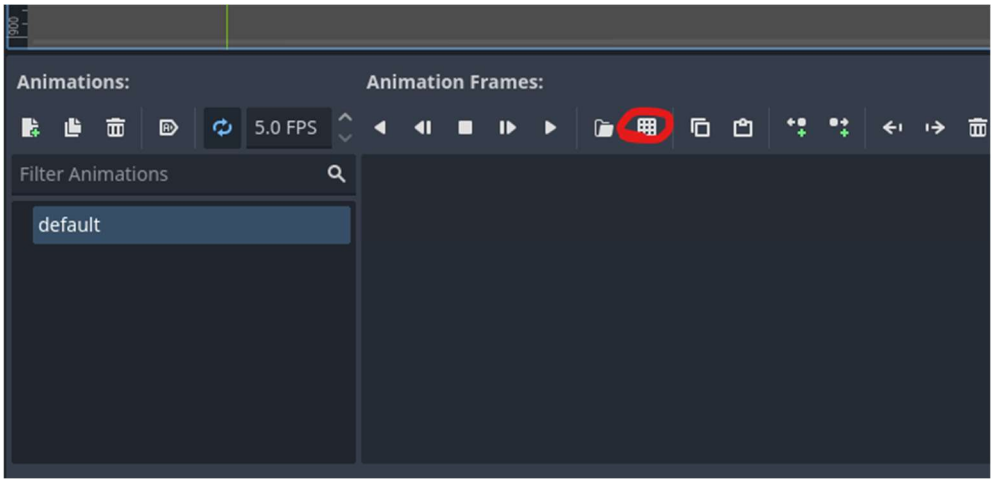


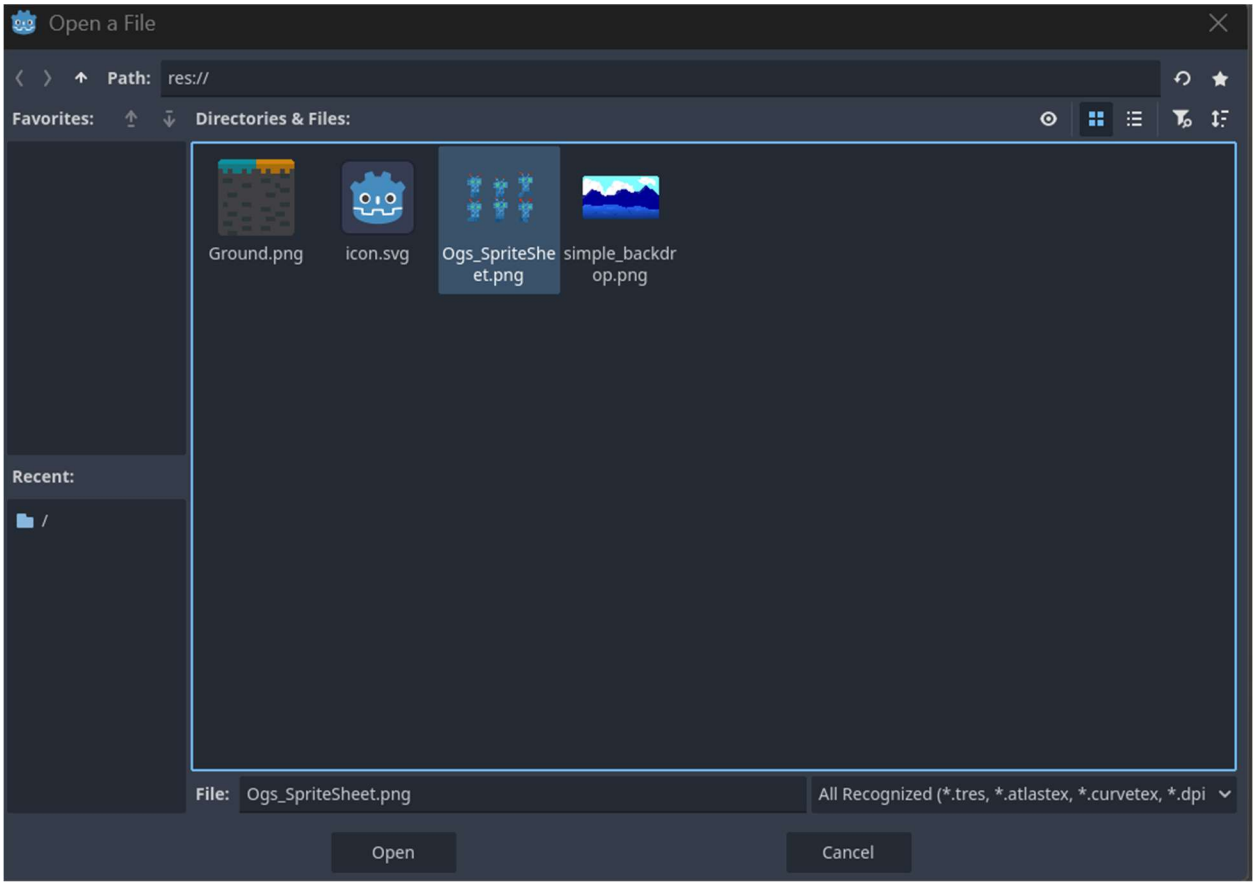


Click on the “Sprite Frames” property name and you should see a new window appear at the bottom.

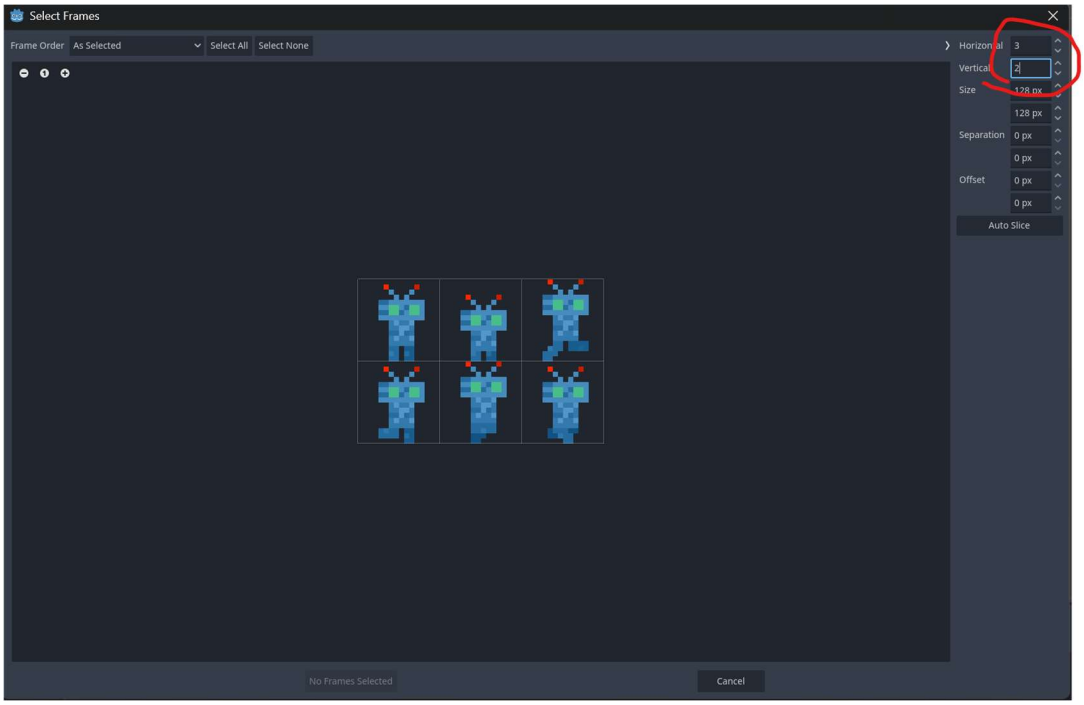


Let’s add the sprite frames now. Click on the grid icon and select the “Ogs_SpriteSheet.png” and then click “Open”.

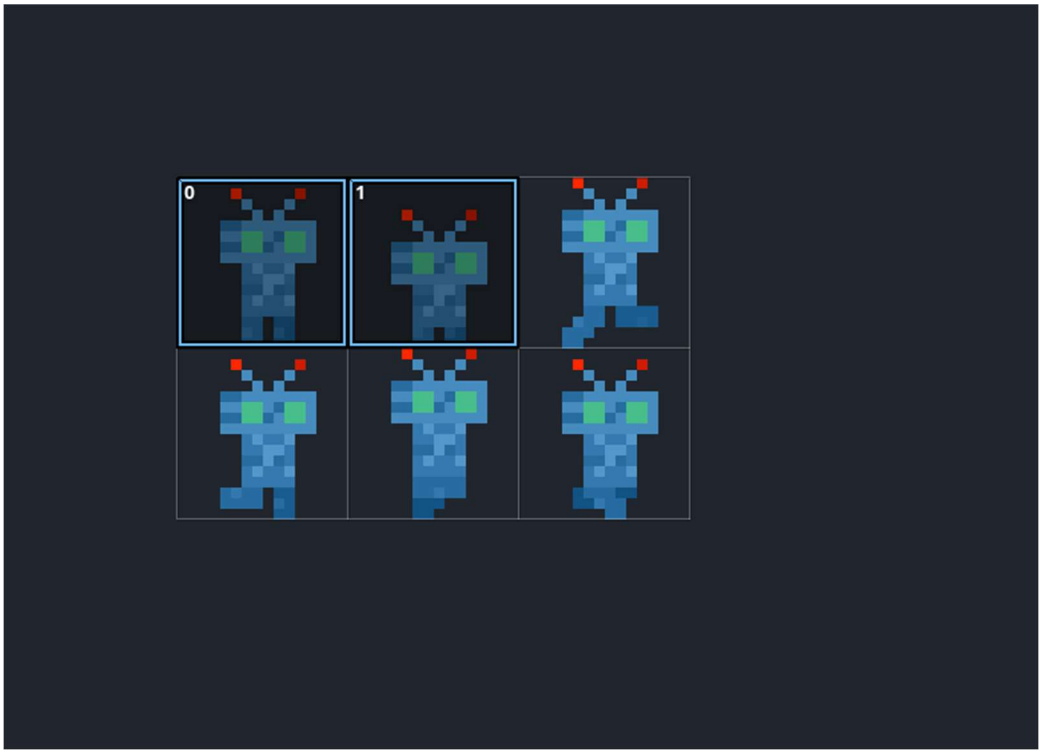




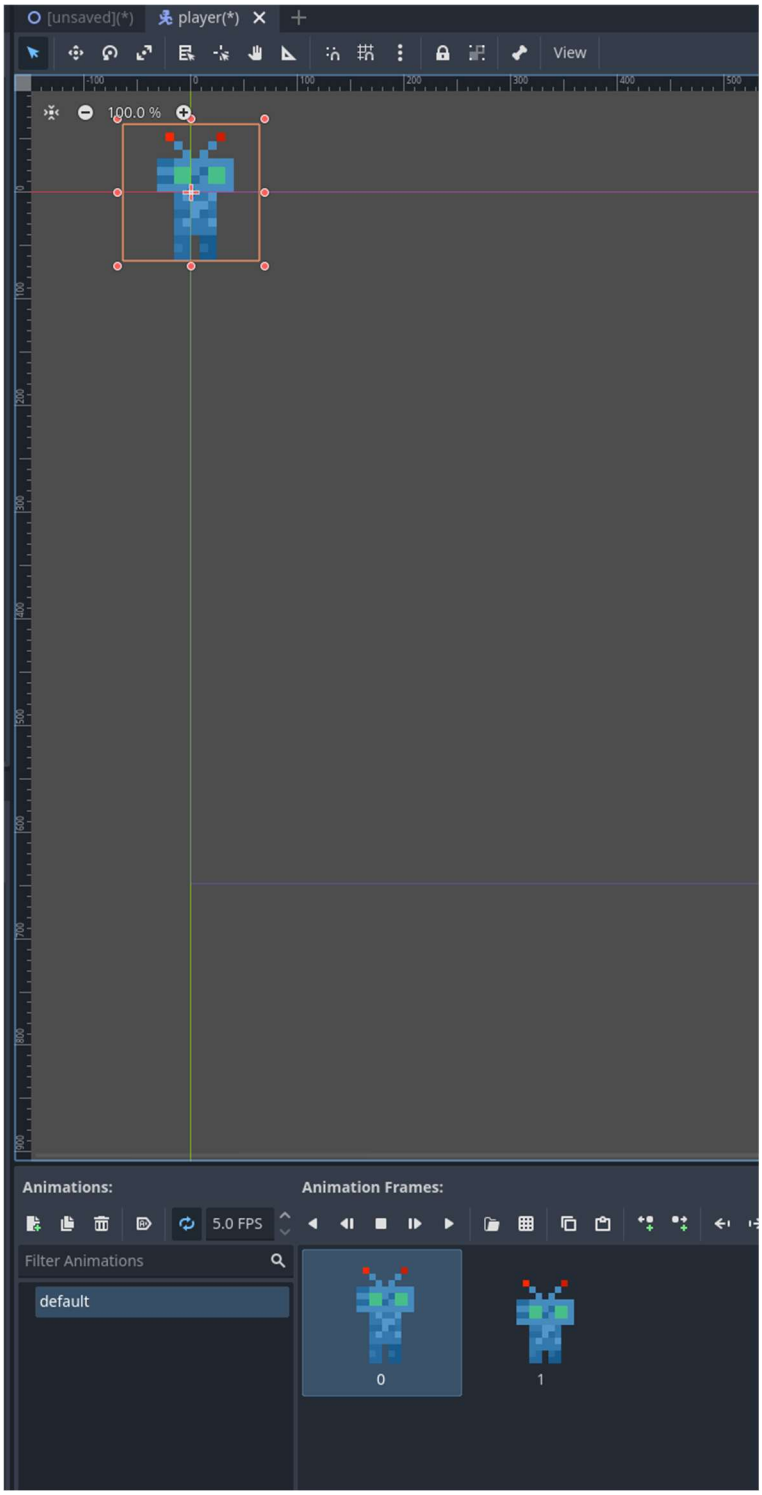
We will now have to let Godot know how big each frame is on the sheet. I exported this png as a 3x2 sprite sheet. Exit the numbers to the right of the window to specify.



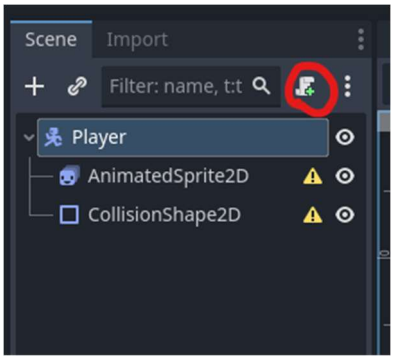
Once the boxes are in the right shape, select the first two as the idle animation.



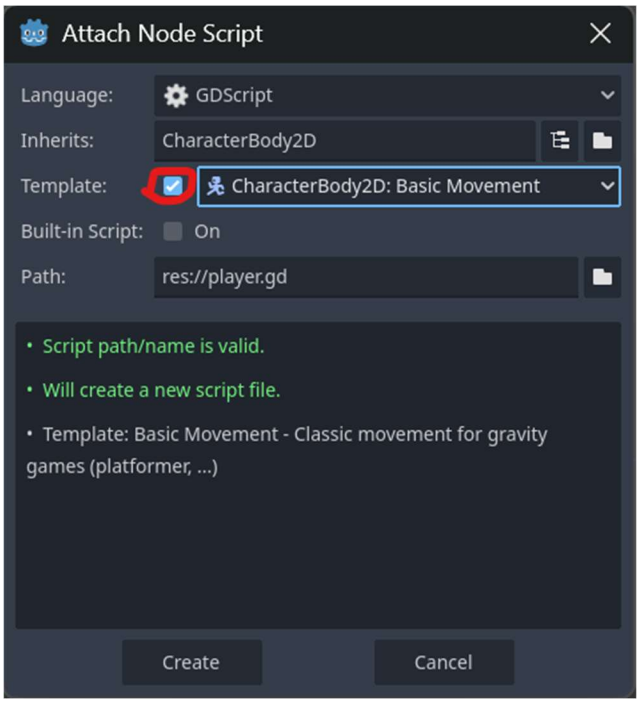
We should now see our player!



We should now add a script to our player node so we can move it. Click on the Player node in the scene selection and click on the icon circled below:



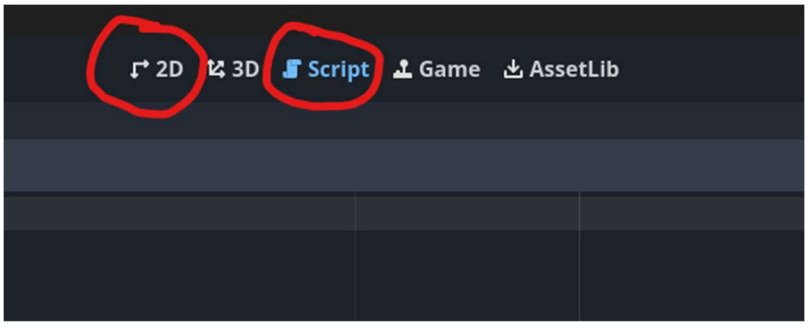
When the Attach Node Script appears, we will want to use the template that comes with the engine, so we don't have to code too much. Click on the checkmark next to Template and add the Basic Movement Script.



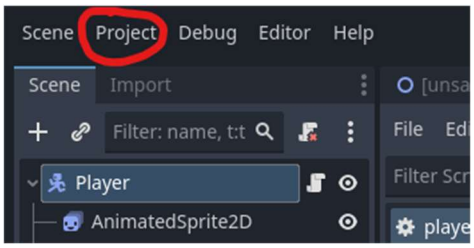
You should now be in the Script editor tab and see the code written in “GD script” which is similar to python.

```gdscript
extends CharacterBody2D


const SPEED = 300.0
const JUMP_VELOCITY = -400.0


func _physics_process(delta: float) -> void:
    # Add the gravity.
    if not is_on_floor():
        velocity += get_gravity() * delta

    # Handle jump.
    if Input.is_action_just_pressed("ui_accept") and is_on_floor():
        velocity.y = JUMP_VELOCITY

    # Get the input direction and handle the movement/deceleration.
    # As good practice, you should replace UI actions with custom gameplay actions.
    var direction := Input.get_axis("ui_left", "ui_right")
    if direction:
        velocity.x = direction * SPEED
    else:
        velocity.x = move_toward(velocity.x, 0, SPEED)

    move_and_slide()

```

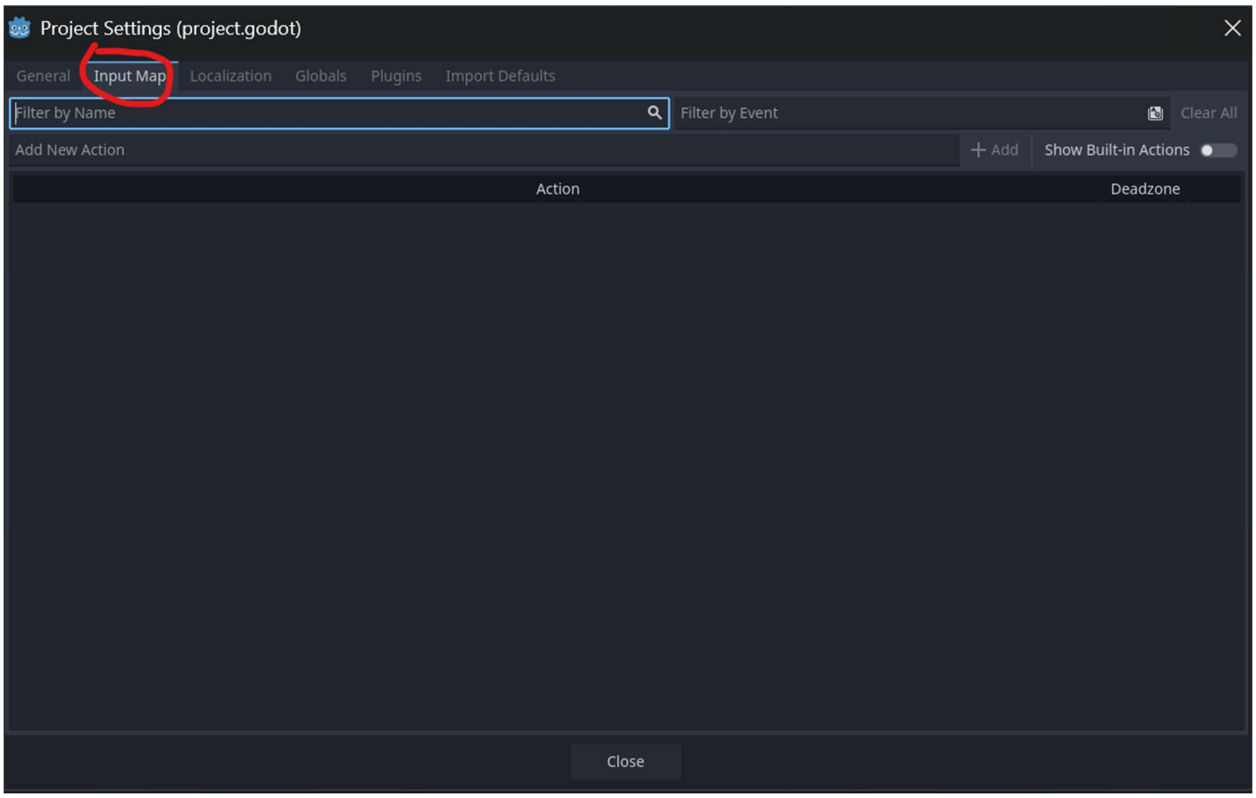
If you ever want to go back to the 2D editor screen you can select which screen you want at the top.



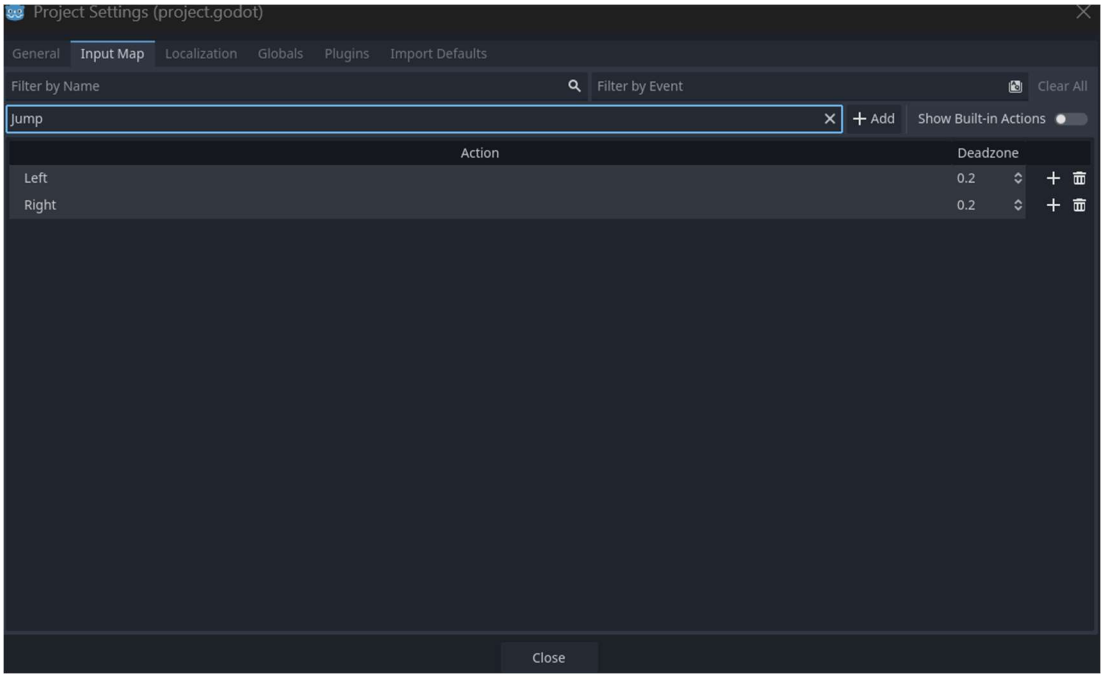
Now, we want to add our custom inputs for movement instead of using the default ones. To do that we need to go into Project Settings.



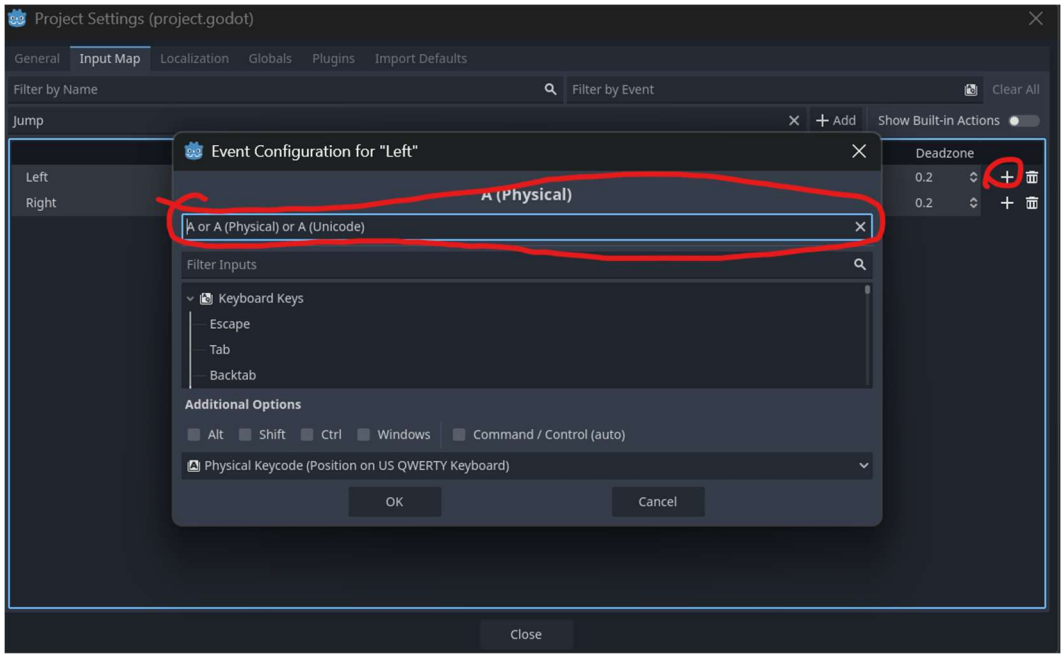
Once there, click on the “Input Map” tab of the pop out window. This is where we can define inputs.



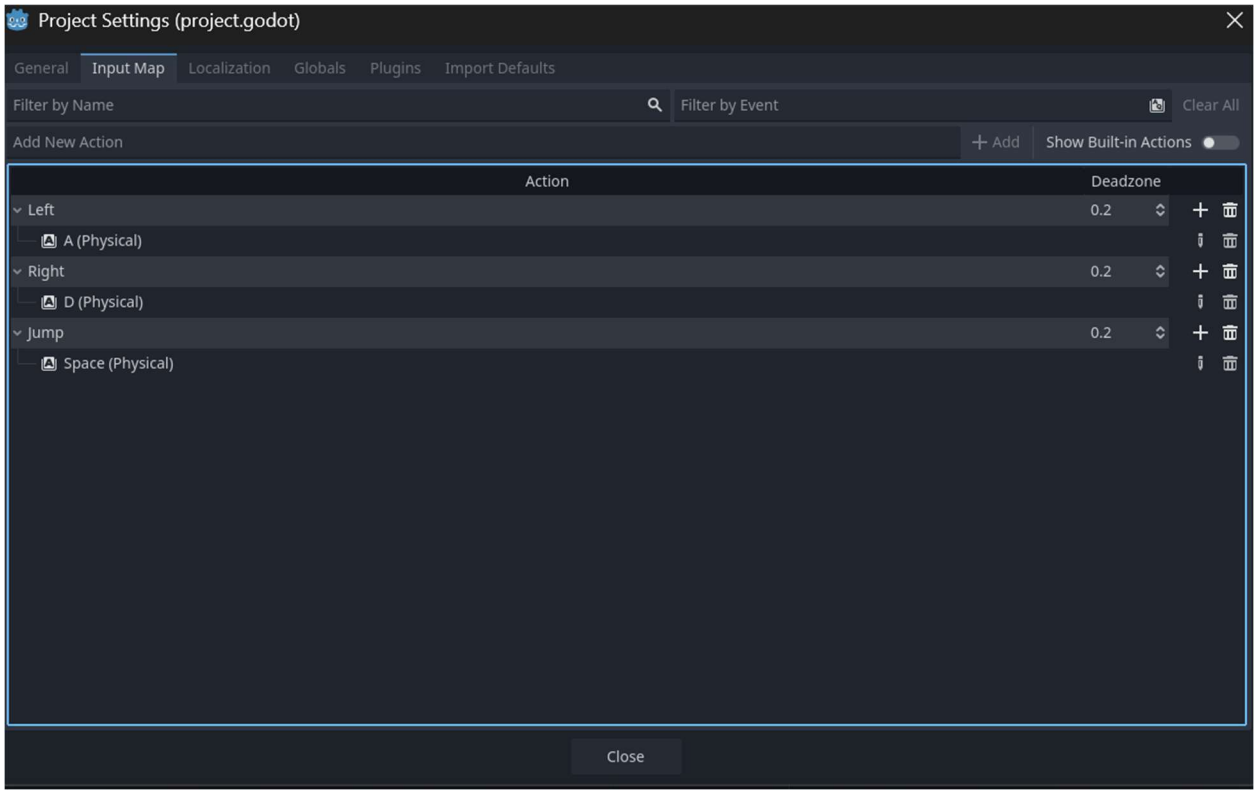
Start adding input labels by typing them in to the box and clicking the “+Add” button, as seen bellow.



Now to map the input labels to the buttons, we will need to press the ’+’ button by each of our labels. On the new pop up we could either look for the buttons we want by going down each list, or by typing the key directly with the box ontop.



Map each keyboard input to your desired configuration for each input lable.

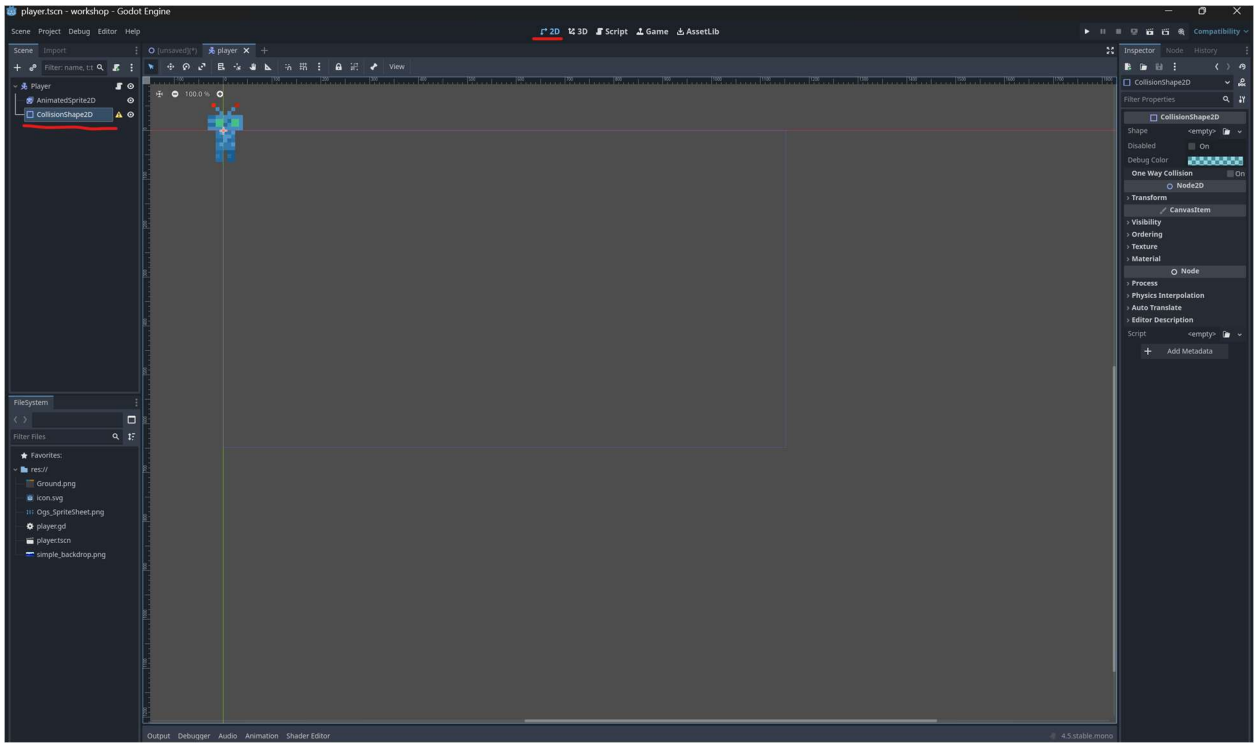


Now we just have to change the labels from the default in the script to the ones we chose:

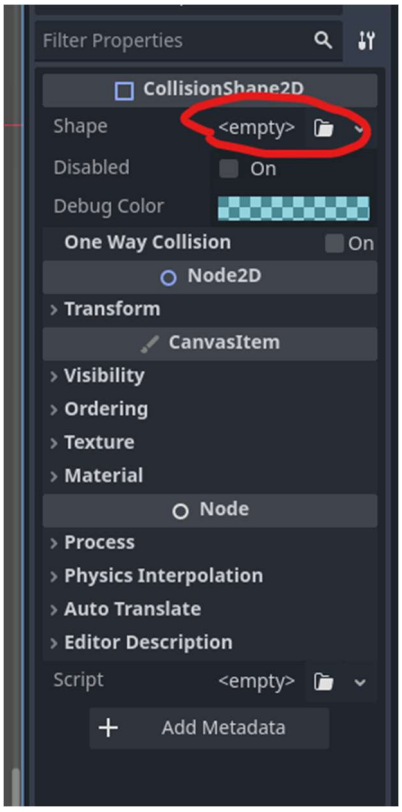
```
7
8  func _physics_process(delta: float) -> void:
9      # Add the gravity.
10     if not is_on_floor():
11         velocity += get_gravity() * delta
12
13     # Handle jump.
14     if Input.is_action_just_pressed("ui_accept") and is_on_floor():
15         velocity.y = JUMP_VELOCITY
16
17     # Get the input direction and handle the movement/deceleration.
18     # As good practice, you should replace UI actions with custom gameplay actions.
19     var direction := Input.get_axis("ui_left", "ui_right")
20     if direction:
21         velocity.x = direction * SPEED
22     else:
23         velocity.x = move_toward(velocity.x, 0, SPEED)
24
25     move_and_slide()
```

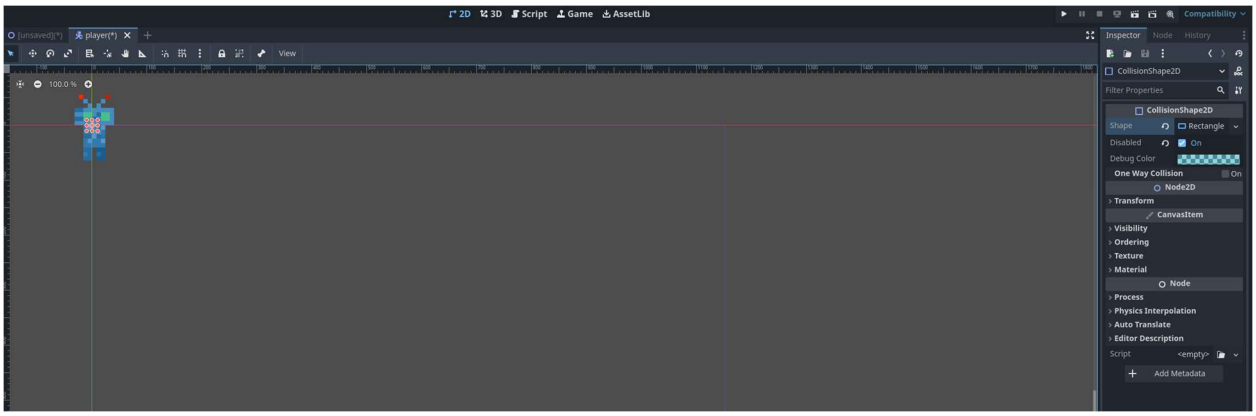
```gdscript
func _physics_process(delta: float) -> void:
    # Add the gravity.
    if not is_on_floor():
        velocity += get_gravity() * delta

    # Handle jump.
    if Input.is_action_just_pressed("Jump") and is_on_floor():
        velocity.y = JUMP_VELOCITY

    # Get the input direction and handle the movement/deceleration.
    # As good practice, you should replace UI actions with custom gameplay action
    var direction := Input.get_axis("Left", "Right")
    if direction:
        velocity.x = direction * SPEED
    else:
        velocity.x = move_toward(velocity.x, 0, SPEED)
```

Now we will need to give our player collision. To do that go back to the 2D editor screen and click on the “CollisionShape2D”.

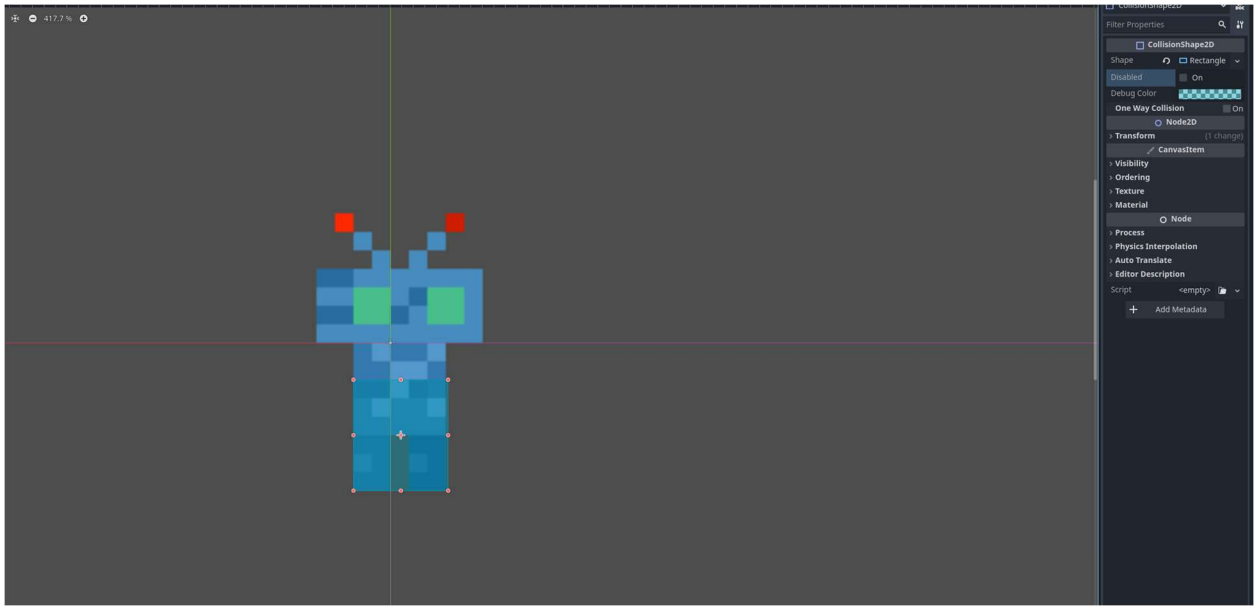


On the right side of the property editor, click the “<empty>” button on the Shape and add a “Rectangle Shape 2D”

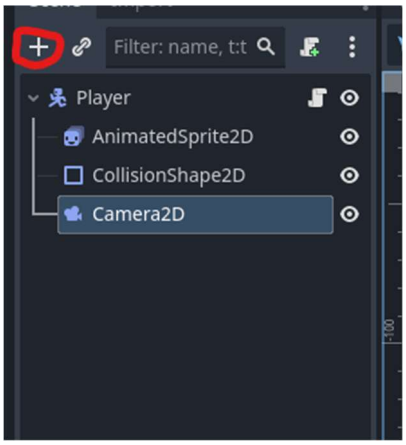




I should mention now that you can move your screen by clicking and holding the middle mouse button. Move over to look closer at your player and then adjust the blue box to fit the sprite. The box is the collision shape and how your character will interact with the environment.

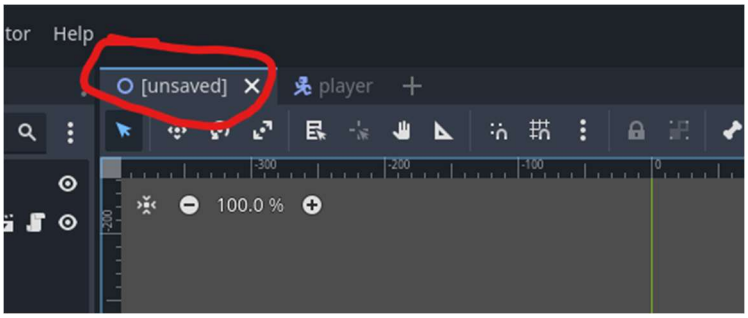


Last thing we need is to add a camera node and we are done with making the Player!

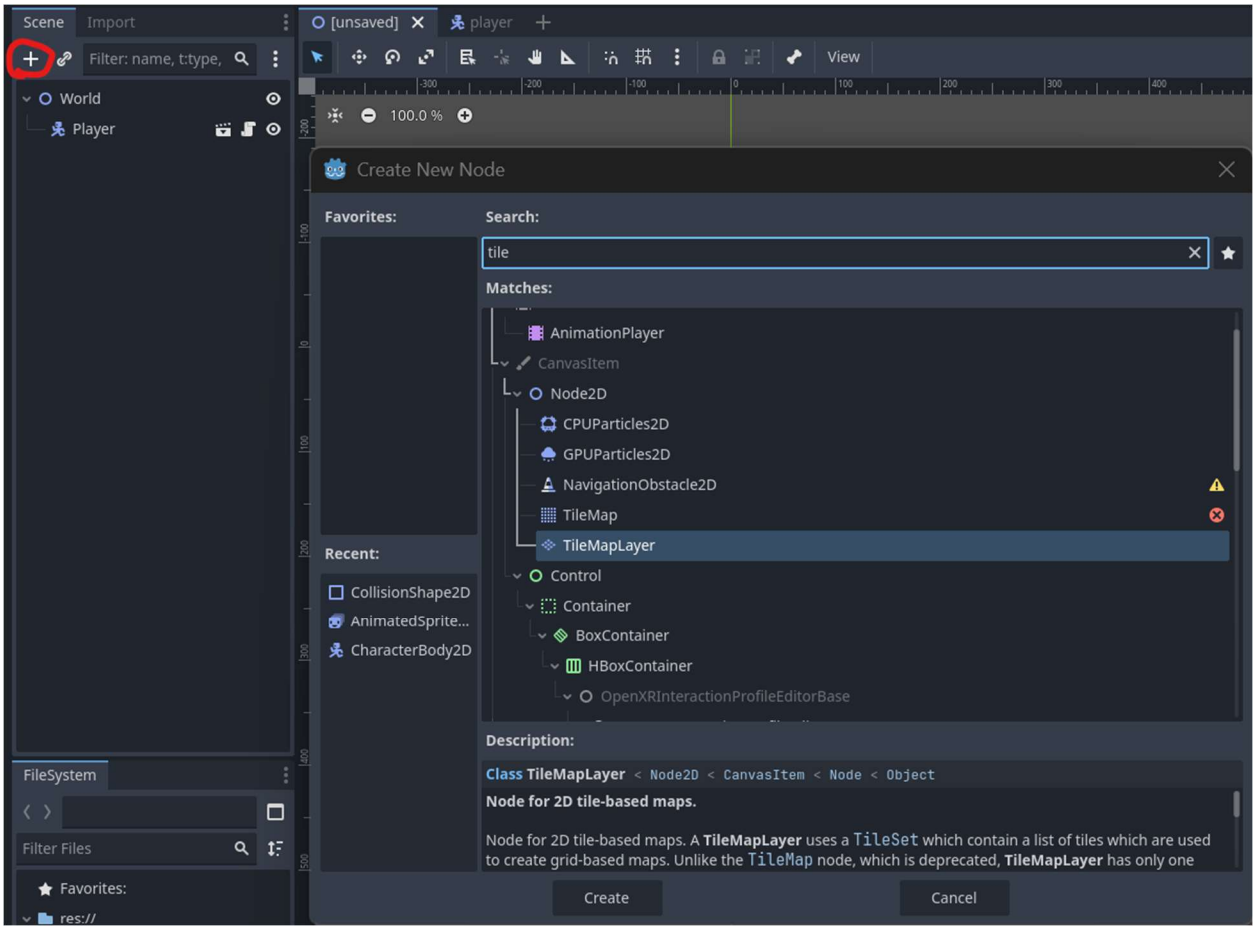


Creating the World

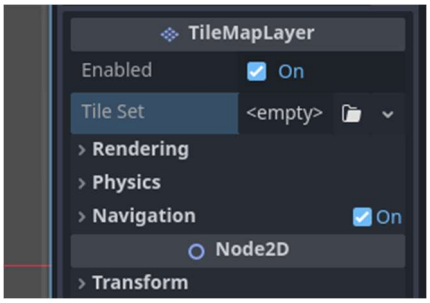
Now we need to make a world that the player can move in. Switch back to our level scene by clicking on the tab.



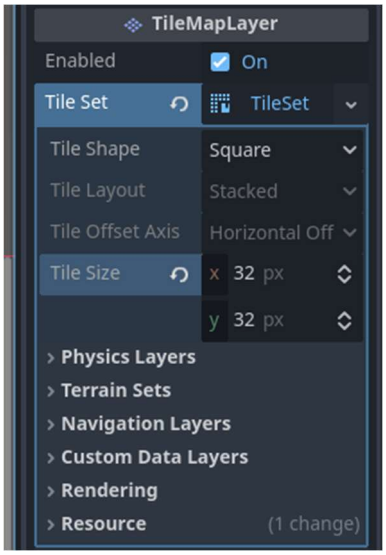
We will need to add a “TileMapLayer” to our scene.



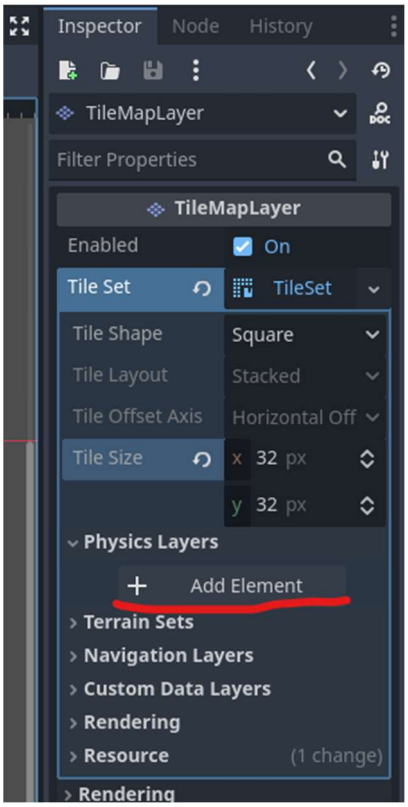
Go the the right side of the screen and once again click on the “<empty>” button for the “Tile Set” Property, and add a new Tile Set.



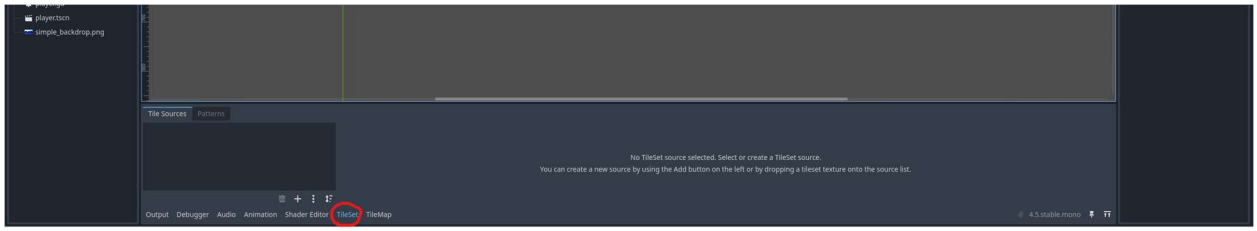
Click on the “TileSet” Button again to edit it’s properties. I drew our environment tiles as 32x32px squares. Edit the “Tile Size” property to 32px.



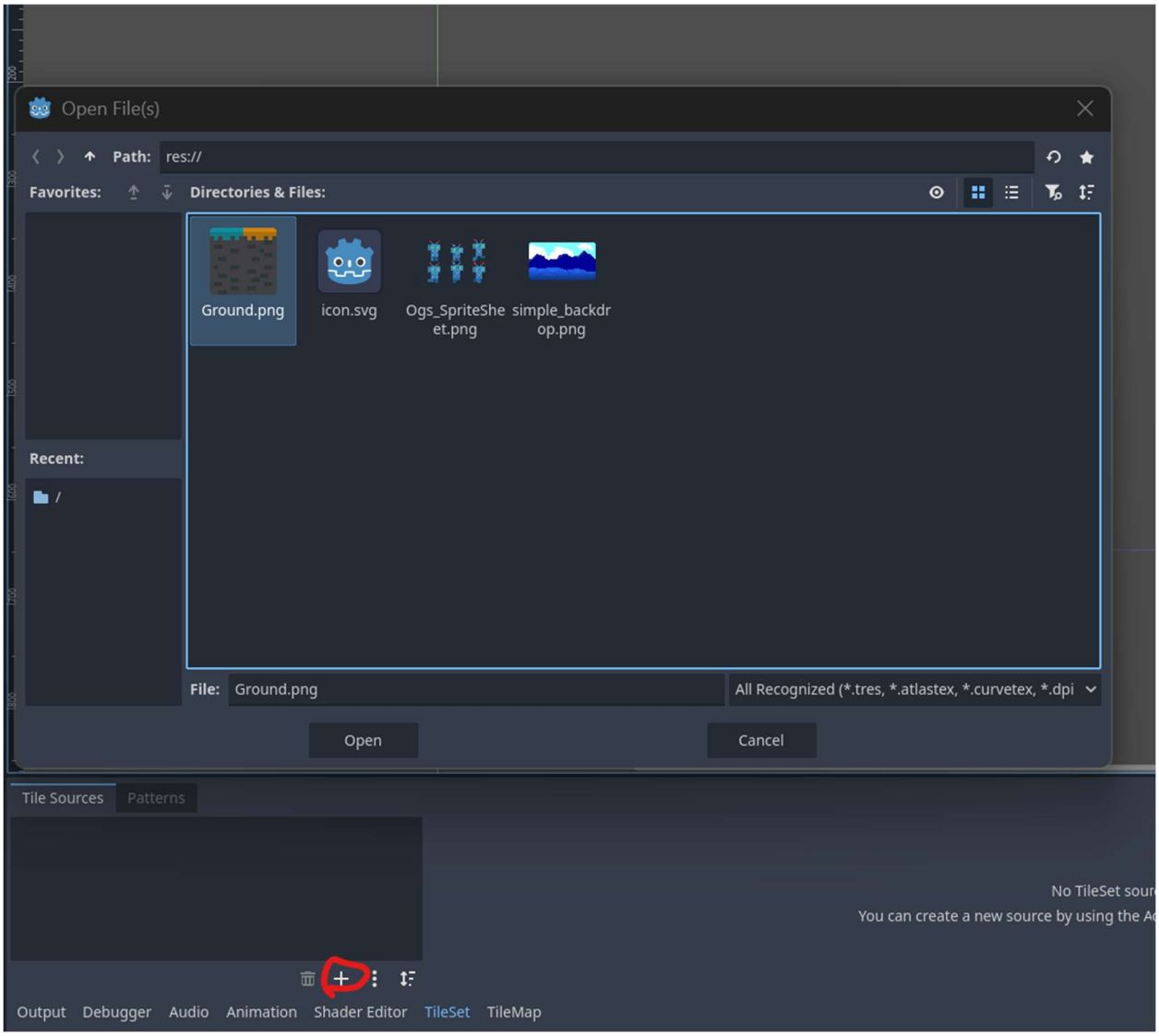
We will need to also add a physics element, so the player doesn't fall through the floor.



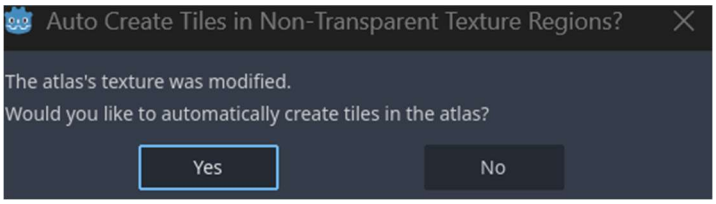
Now we must add a sprite to our tile set. On the bottom screen click on the "TileSet" tab.



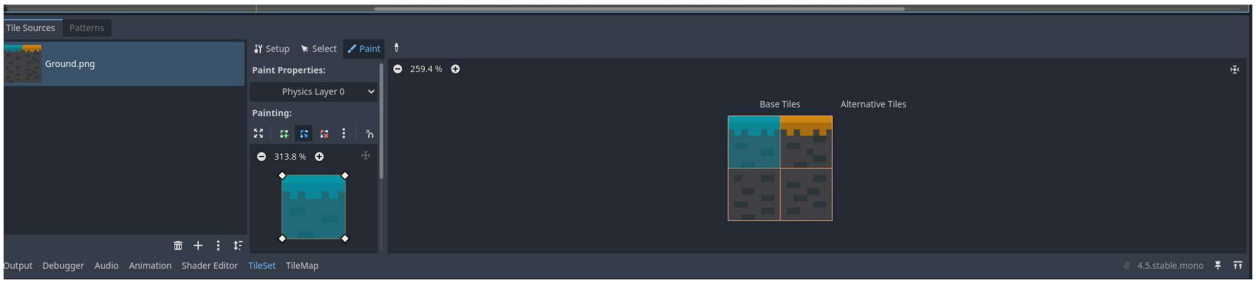
Click on the ‘+’ and click on “Atlas”. Open the “Ground.png”



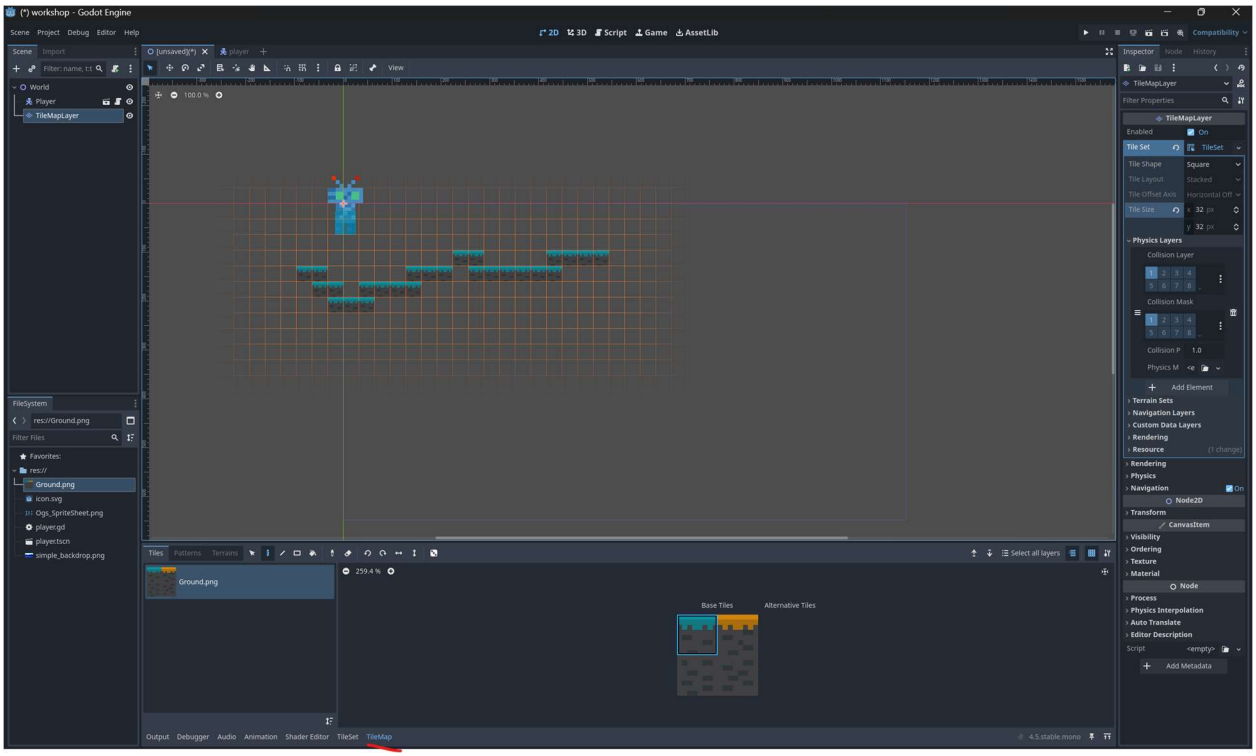
Godot will be able to automatically draw squares on the right tiles, click ‘yes’.



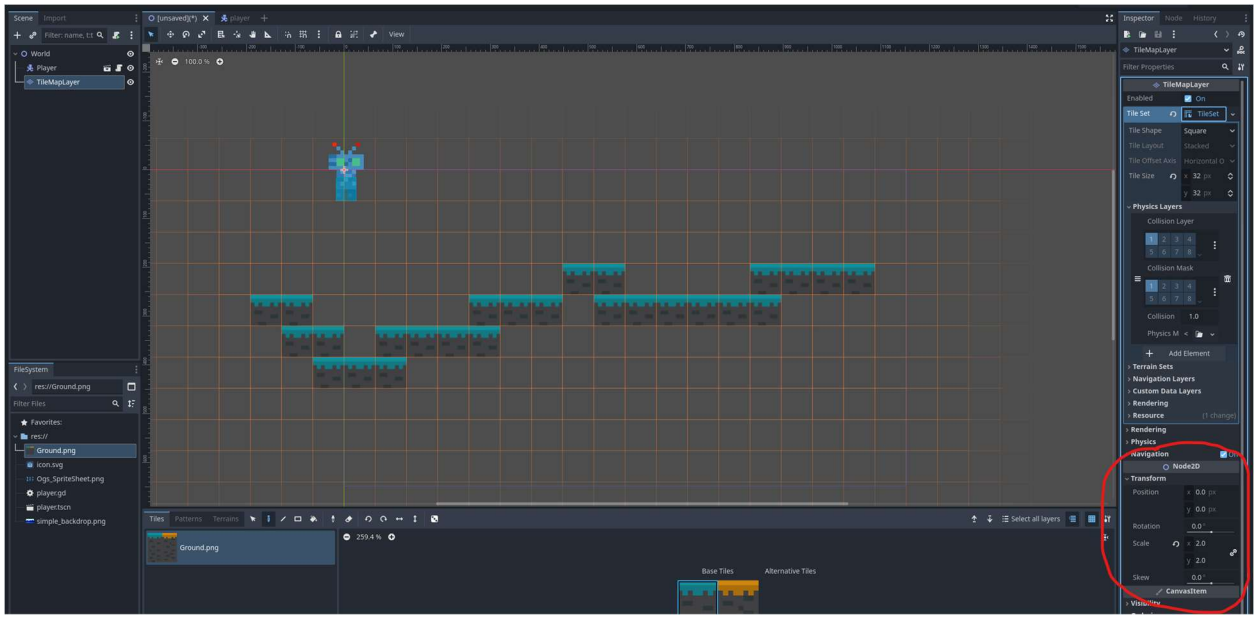
Now we will have to give our tiles collision. Click on the “Paint Button”. And the using the property editor drop down click on “Physics Layer0”. You can now click on each square to give it collision



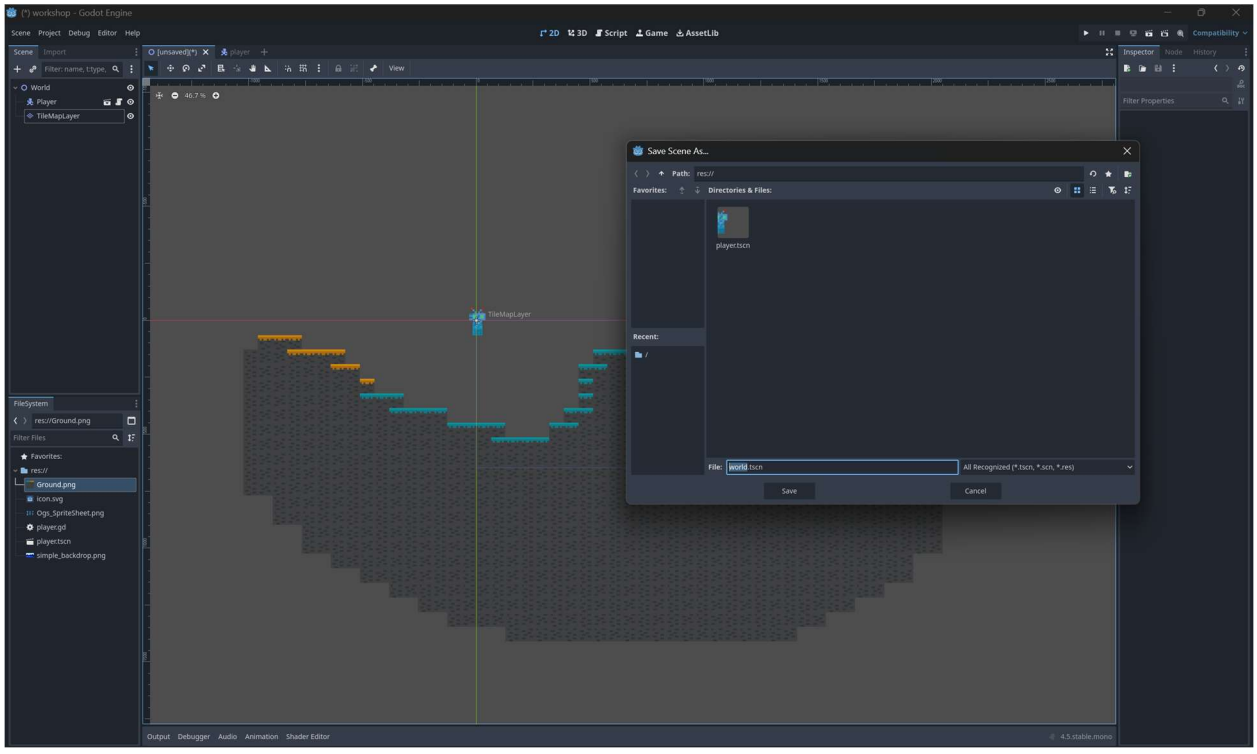
You can now click back on the “TileMap” tab at the bottom of the screen and can now draw a level.



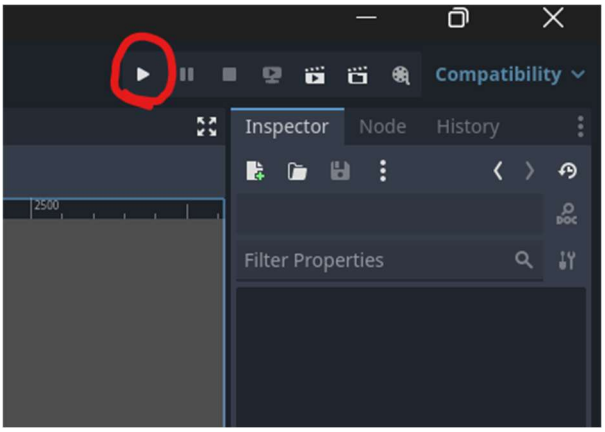
If the tiles are too small you can go over to the properties tab on the right side and increase the scale to 2.



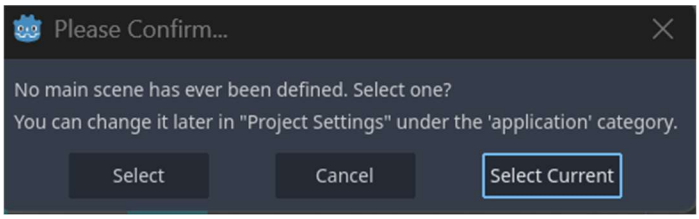
Once you are happy with your level, make sure you save the scene.



Now let’s try playing our level. In the top right of the screen click the play button.



Select Current scene with the pop up.



Congratulations! You are playing a game you made!!