

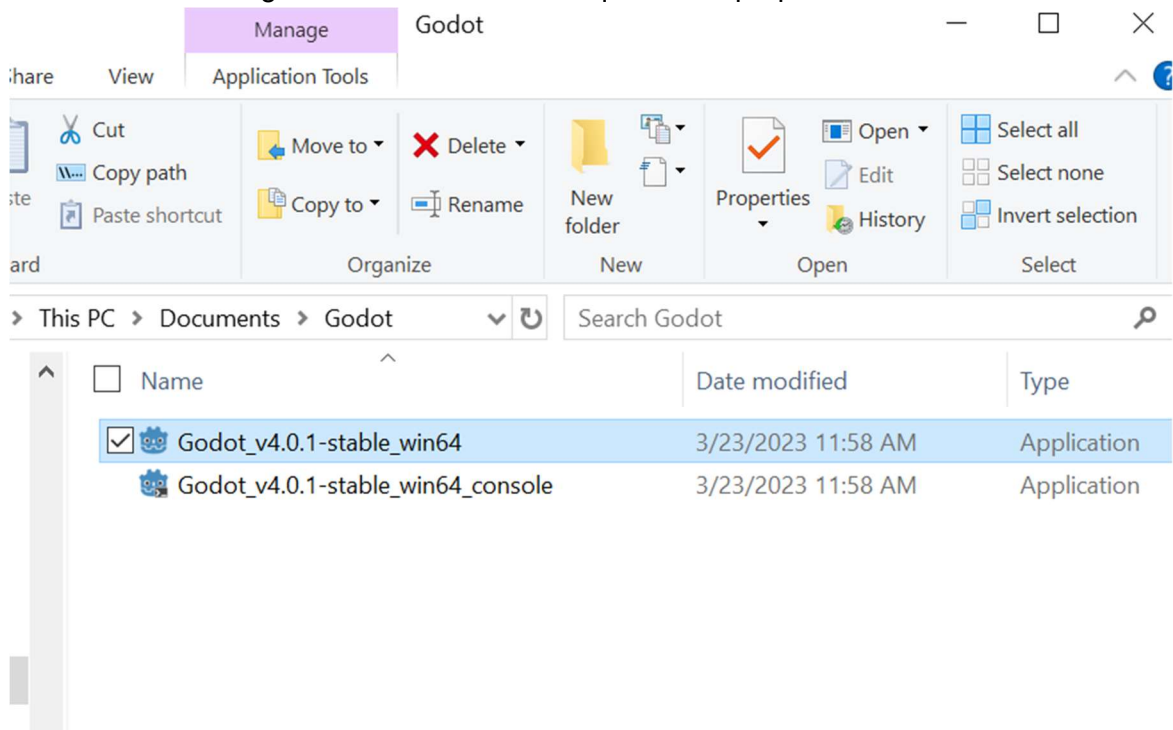
# Godot Workshop Step-By-Step Design Guide

CSU Washkewicz CPT Program

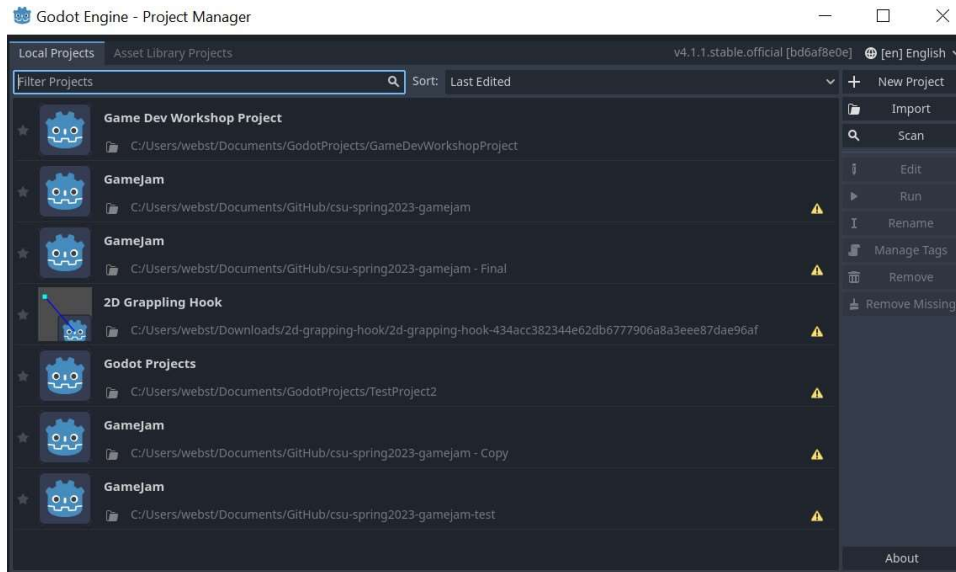
1. Install and run Godot! We'll be using the most recent version of Godot for Windows in this guide (4.1.1) but you can certainly use an older version - just be aware there will be differences in names and UI. Any operating system on which Godot is released is acceptable.



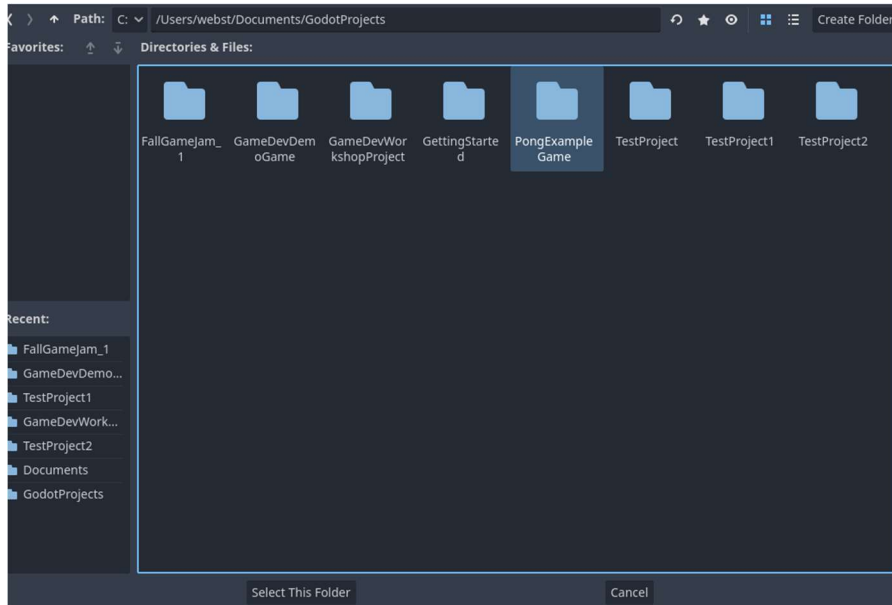
2. Godot runs directly from the executable. To launch Godot, find the .exe file in the directory you installed it to, or search for the executable in your file system. I recommend creating a shortcut to the file for quick-start purposes.



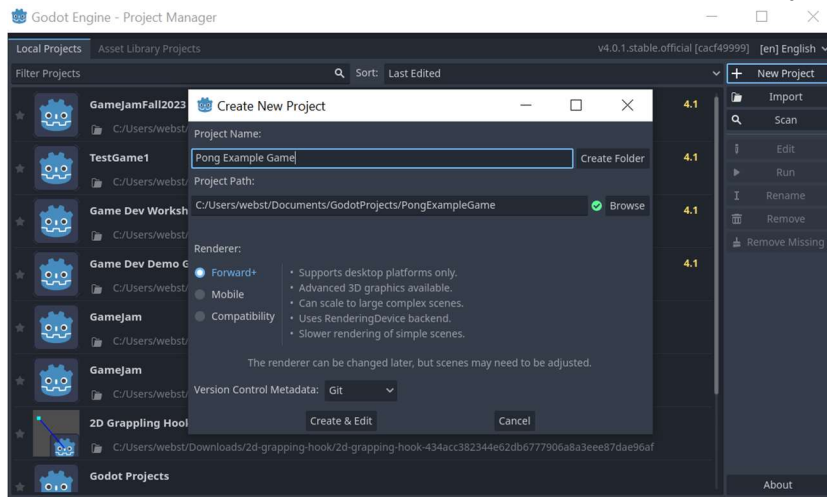
3. Your Godot Client on startup should look like below - just with less projects! To create a new project, click the “New Project” button in the top right. If you’ve used Godot before, but are updating to Godot 4 or 4.1 you’ll see yellow exclamation points on your old projects indicating they need to be updated for the new version of Godot.



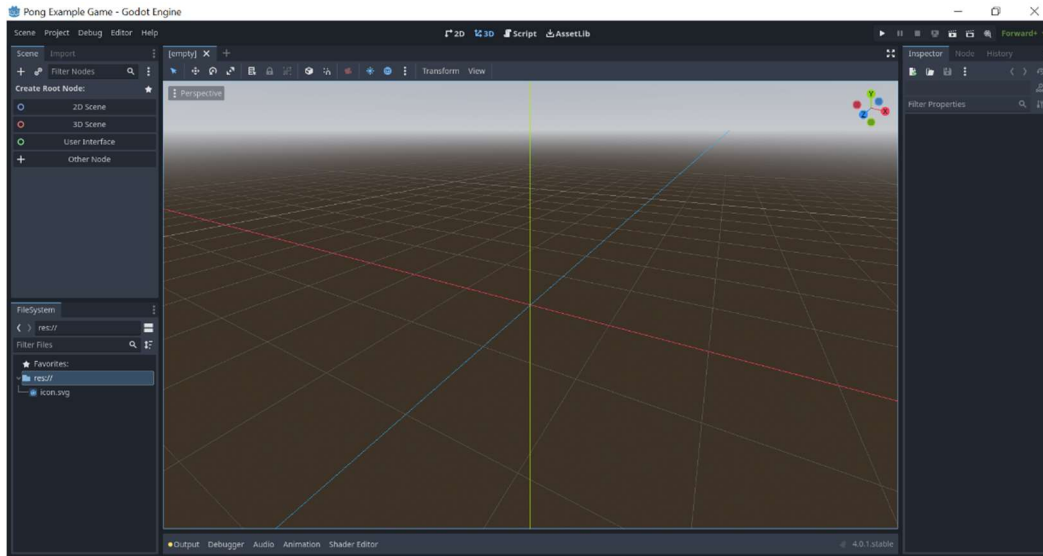
- Rename your new project to “Pong Example Game”. Edit the project path by clicking “browse” and creating a new directory somewhere on your PC to store your Godot Projects; I recommend you name this “Godot\_Projects”. Create a directory inside of Godot\_Projects name “PongExampleGame” to store all the files and assets for our game. Ensure you are inside of this directory and click “Select This Folder”.



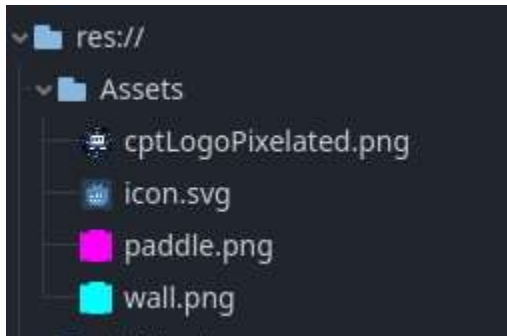
- The Renderer options allow you to choose more or less advanced renderers which are compatible with different platforms. For this example you can select Forward+, though you may need different options for different projects. Leave the Version Control Metadata option on Git and click “Create and Edit” to create your game!



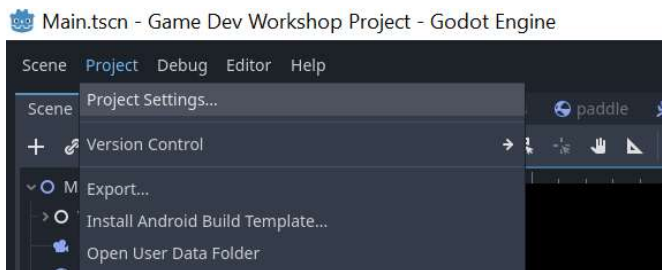
- You should now see the Godot Game Engine’s UI. We will walk through the various parts of this during the guide, but for now examine the FileSystem tab in the bottom left of your screen. There should be a folder named “res://” and within it a single svg file called “icon”.



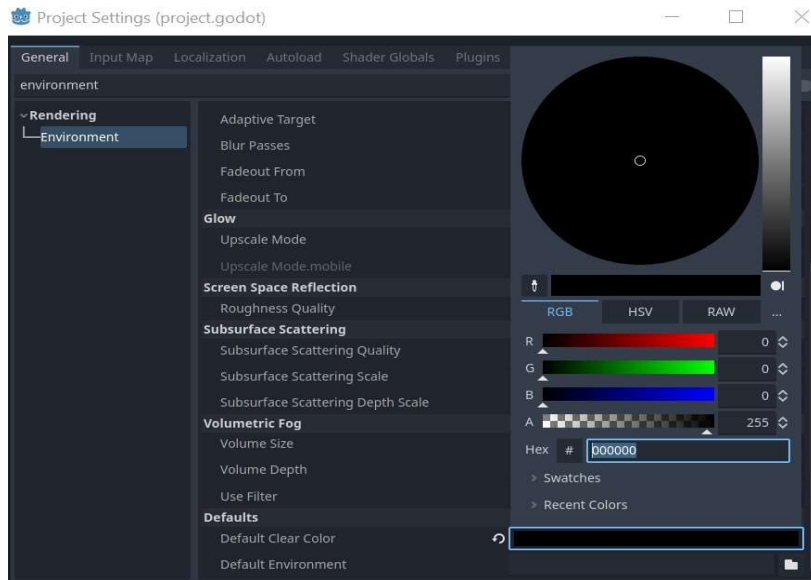
- Download the premade GameDevWorkshopAssets folder from the CPT Github and extract the contents. Create a new folder by right-clicking the "res://" folder in the FileSystem tab and selecting "new" -> "folder". Rename this folder "Assets". Drag the files from the extracted folder into the newly created Assets folder as shown below.



- Click on the Project tab in the top left and open the Project Settings window.

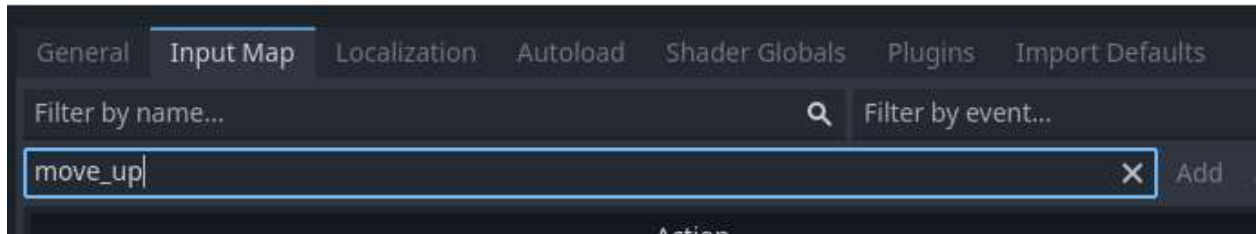


Enter "environment" in the Filter Settings searchbar at the top of the window and click on the Environment tab on the left. Scroll down to the Defaults category and change the Default Clear Color to 0x000000 (Black). This allows you to modify the standard background color for your scenes.

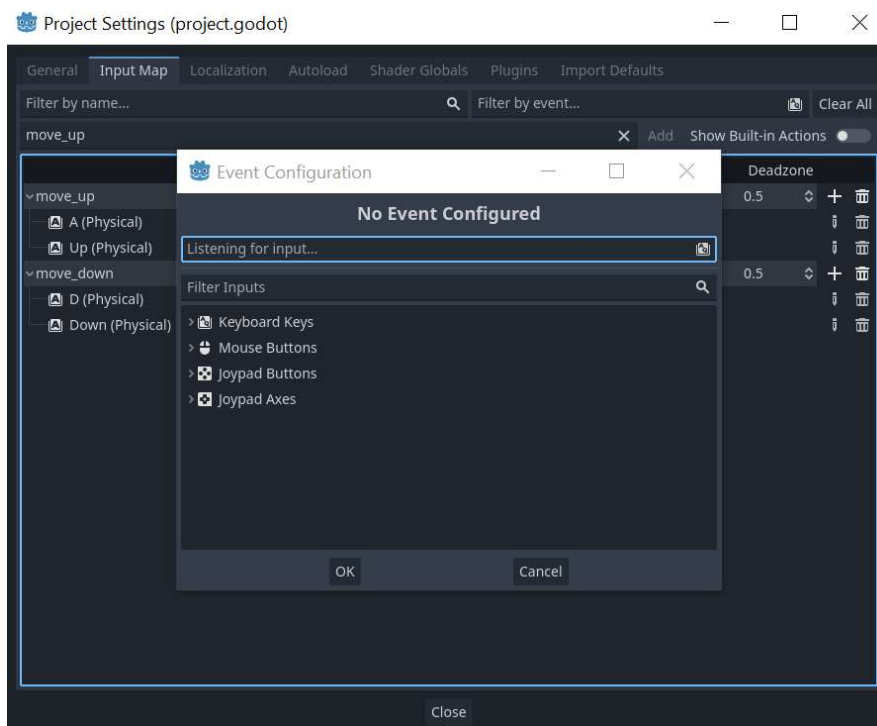


- Click on the “Input Settings” tab at the top of the window and type “move\_up” into the Add New Action bar. Click Add to add this action to your list of game actions.

### Project Settings (project.godot)

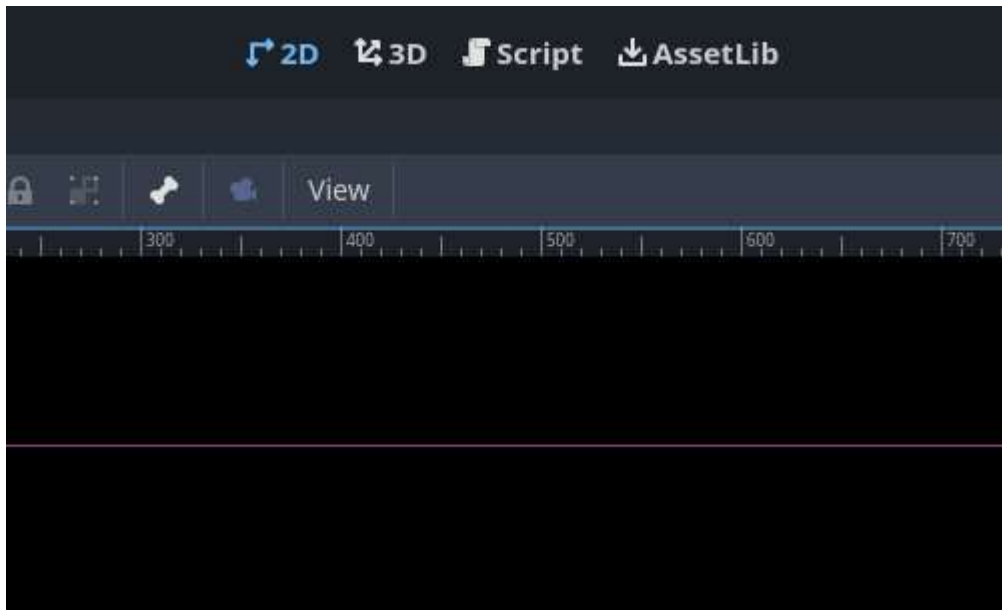


Click the Add Event button on the right of your new action to open the Event Configuration Window. This allows you to create set inputs for certain actions in your game. We'll be adding the “a” and “up” keys to this action.

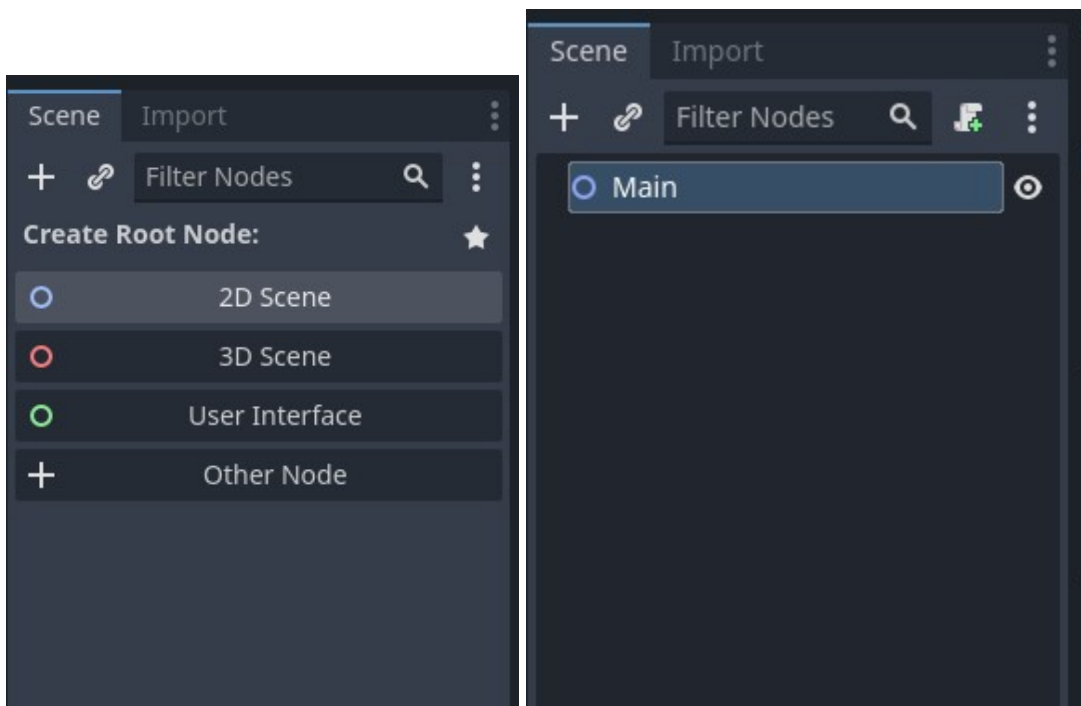


Press the “a” key and click OK to add the key to “move\_up”. Add another event by clicking the Add Event button again; this time, press the “up” key and click OK. Repeat this process to create a “move\_down” action with the “d” and “down” buttons bound to it. You can now close the Project Settings window.

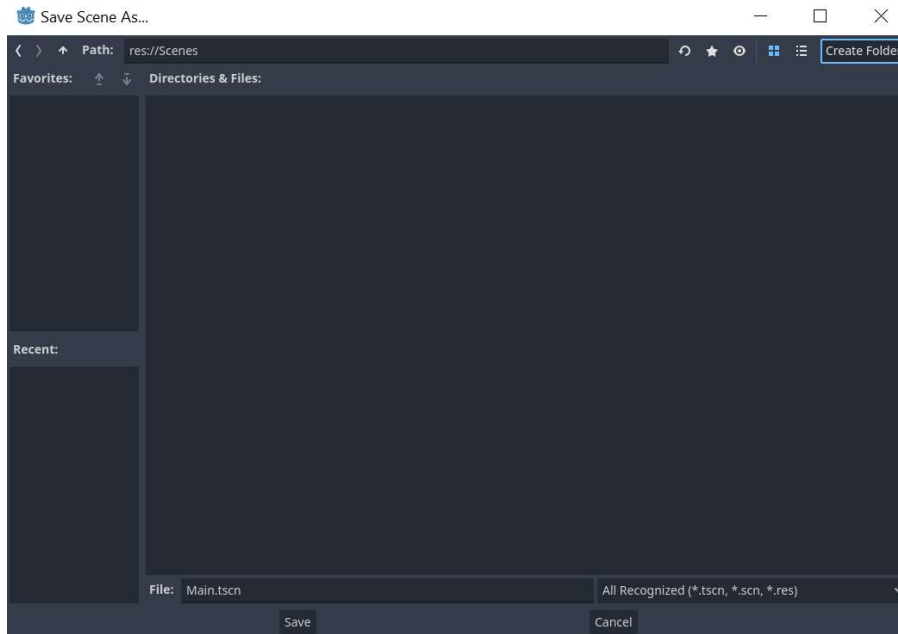
10. Switch from the 3D view to the 2D view by clicking “2D” above the game window.



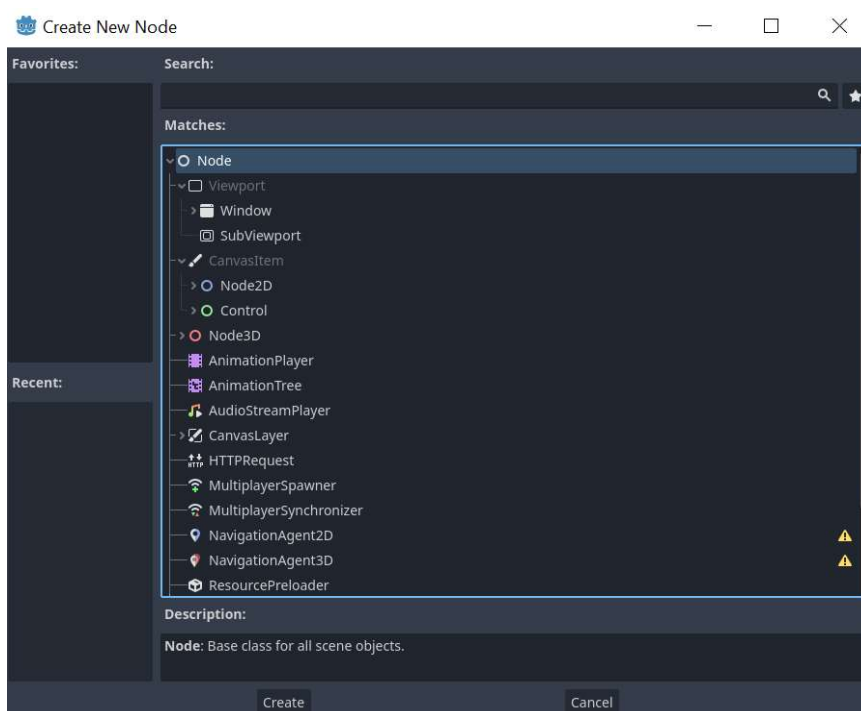
11. Under the Scene tab on the left side, click the 2D scene button to add a new scene. This will serve as our main scene for the project and the root node of all other nodes. Rename this scene “Main”.



Press “ctrl+S” to save the scene and open the “Save Scene As” window. Create a new folder and name it Scenes; save this Main scene within that folder. Be sure to rename the scene “Main.tscn” - it is proper Godot convention to capitalize scene names.



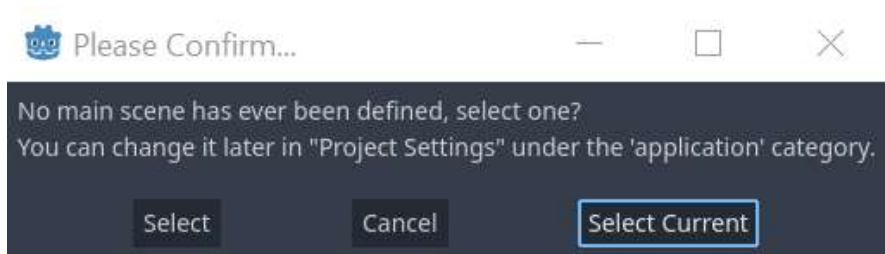
12. In this Main scene, create a child node by right clicking and selecting “Add Child Node” or by pressing “ctrl+A”. This will open the “Create New Node” Dialogue - from here, you can view the multitude of nodes Godot has to offer. Select the generic Node node (likely the first option selected) and click Create.



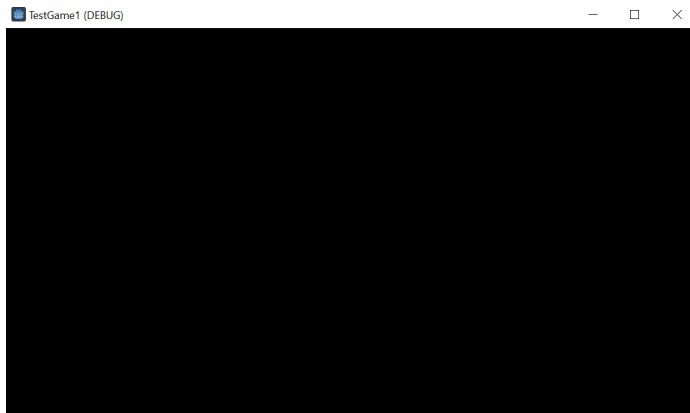
This node will act as a “container” for our various wall nodes that make up the level. As such, rename the node Walls and save it.



13. It's time to check our progress so far and to select our main scene for the project. Make sure the "Main" scene we created first is selected and press F5 to run the project.

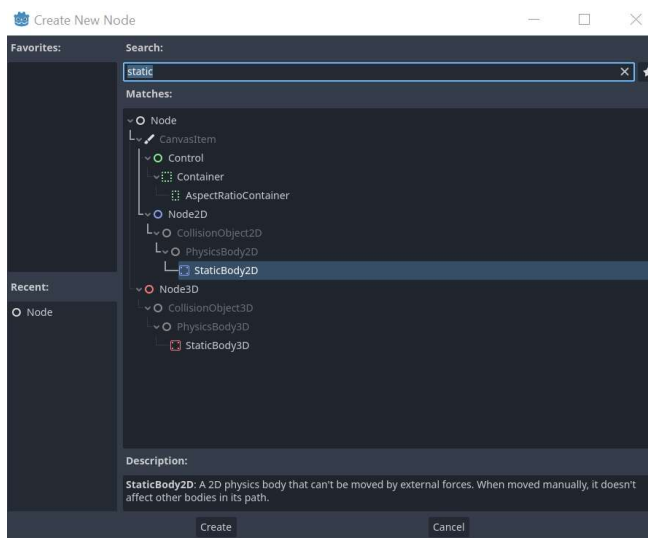


A dialogue will pop up asking you to define a main scene. Click Select Current if you have the "Main" scene selected; this can be changed later if necessary.



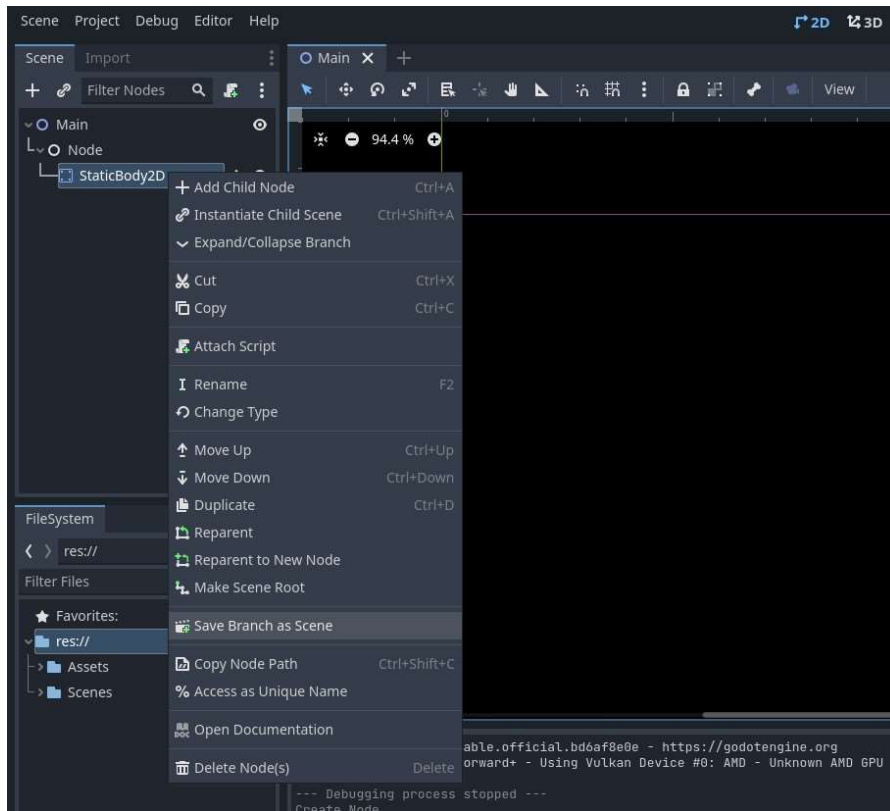
A blank black window should pop up with your ProjectName(DEBUG) as the title; this is your current game! As we add more nodes and scenes, it will slowly take shape as a finished product. You can now click the X to close this window, having tested so far.

14. Within the new Walls node, create a new node. Select the type to be StaticBody2D; you can do this by searching the type in the searchbar at the top of the window.

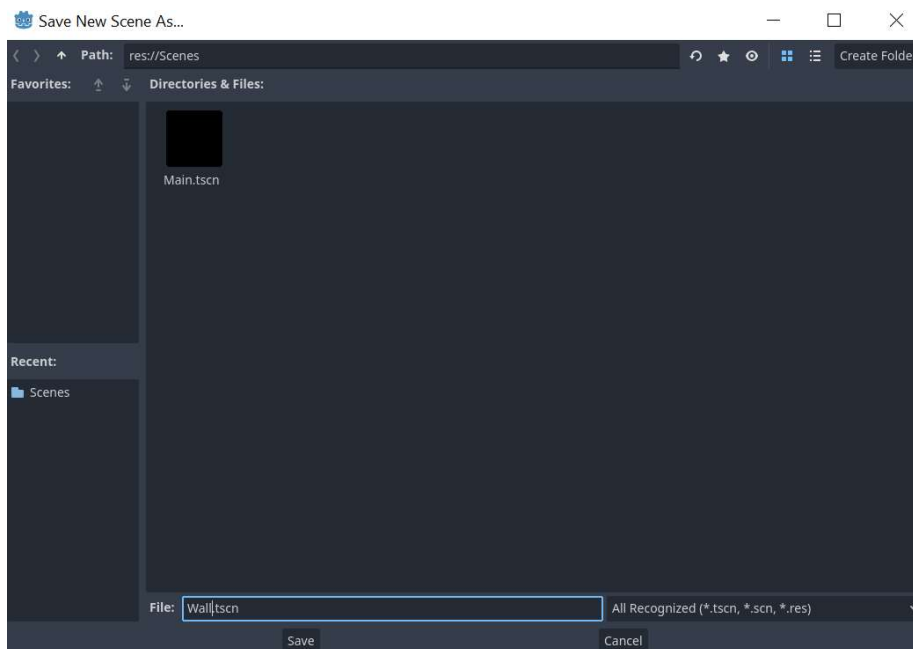


Select the StaticBody2D and click Create.

15. This node will serve as the root for our Wall scene; as such, right click on the StaticBody2D node and select “Save Branch As Scene”.

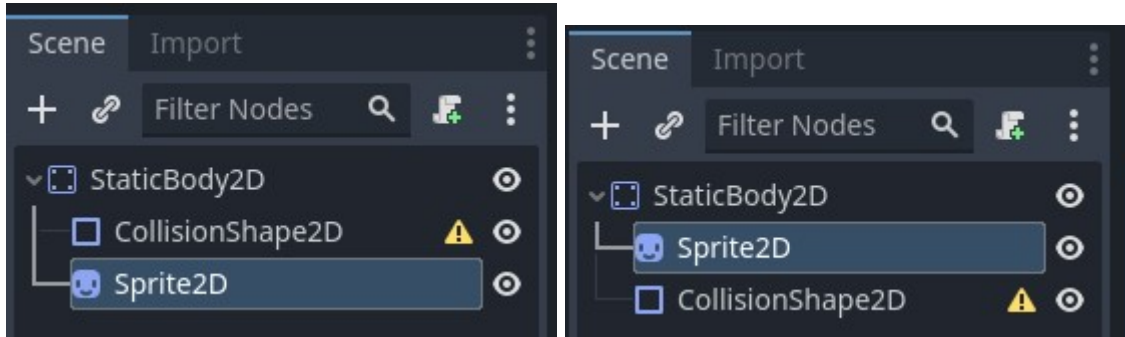


In the “Save New Scene” window, save the scene in the Scenes folder as “Wall.tscn”.



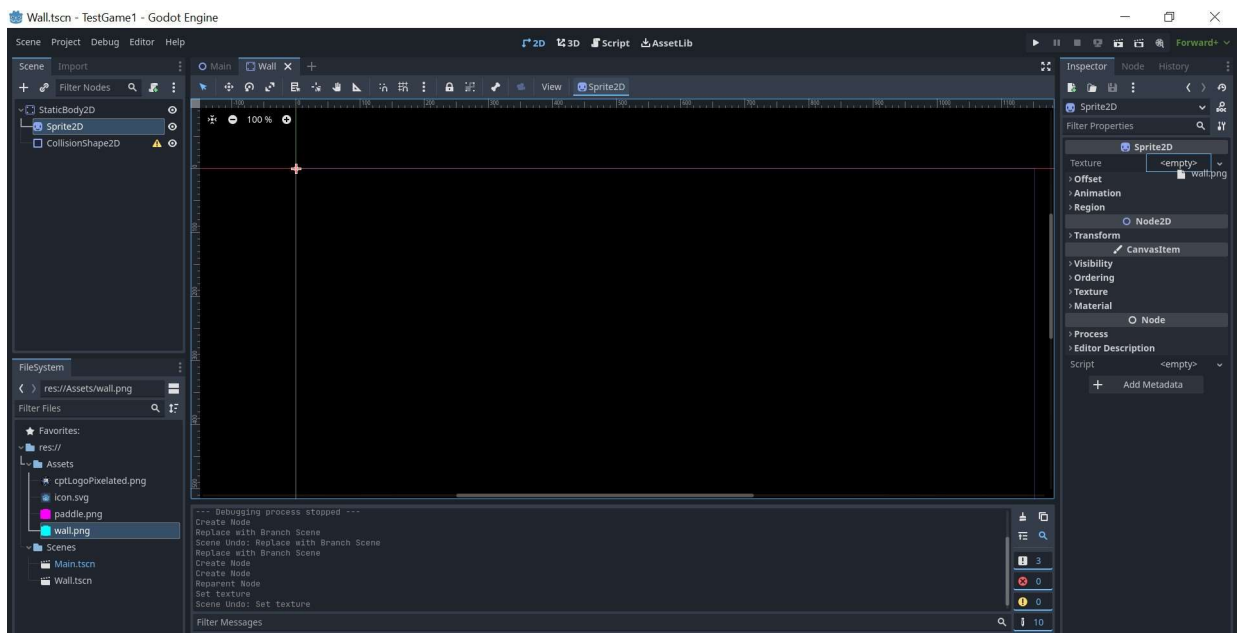
16. Double-click on the Wall scene within the FileSystem tab to open the scene - also notice that you can view each scene as a tab at the top of the Viewport window. In the

StaticBody2D node, create two new nodes: a CollisionBody2D node and a Sprite2D node. Ensure that both of these nodes are children of the StaticBody2D node and are at the same level in the node hierarchy. Drag the Sprite2D node above the CollisionBody2D node in the hierarchy; this will allow us to view both nodes simultaneously while the Sprite2D node is selected.

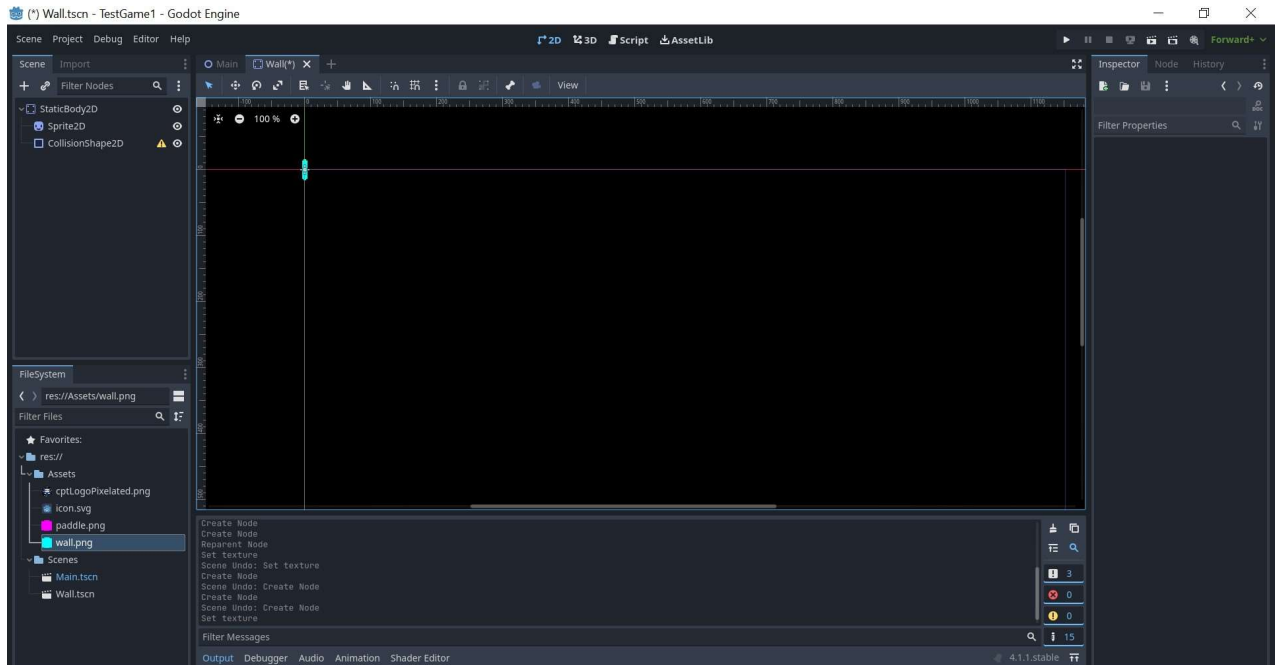


You'll notice a yellow exclamation point next to the CollisionShape2D node; this is normal, and we will fix this error shortly.

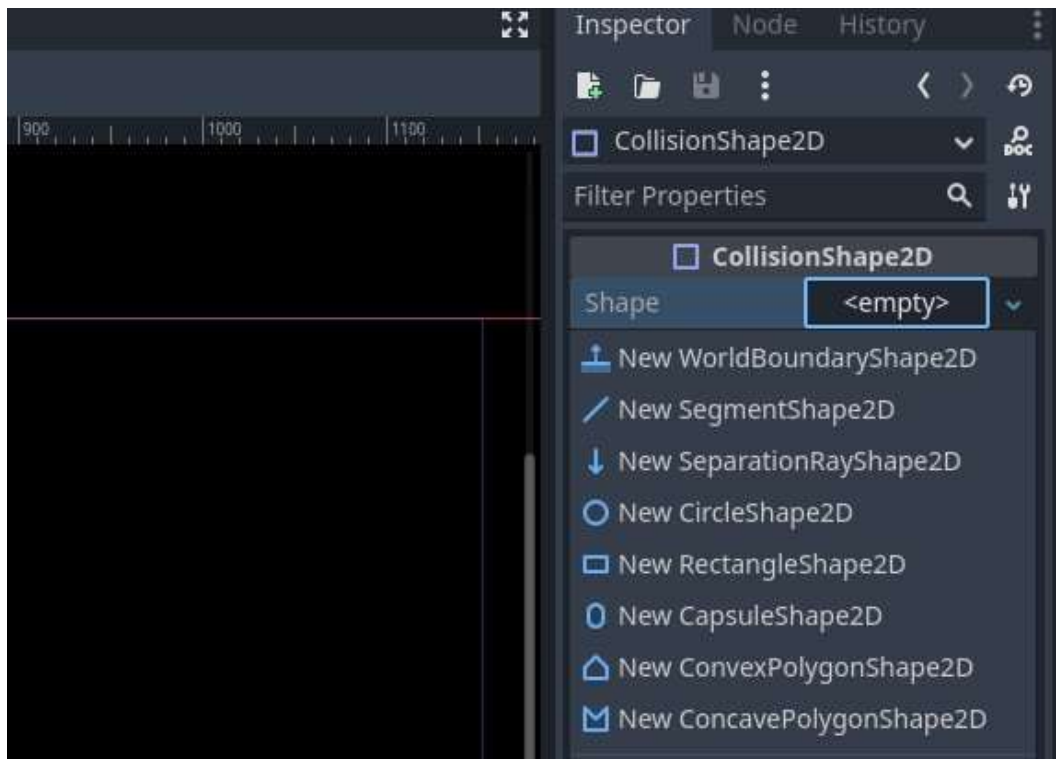
17. Select the Sprite2D node. Note the Inspector tab on the right side of the viewport; it should have Sprite2D listed. Open the Assets folder in the Filesystem window and drag the 'wall.png' file to the texture box in the Inspector tab and drop it in.



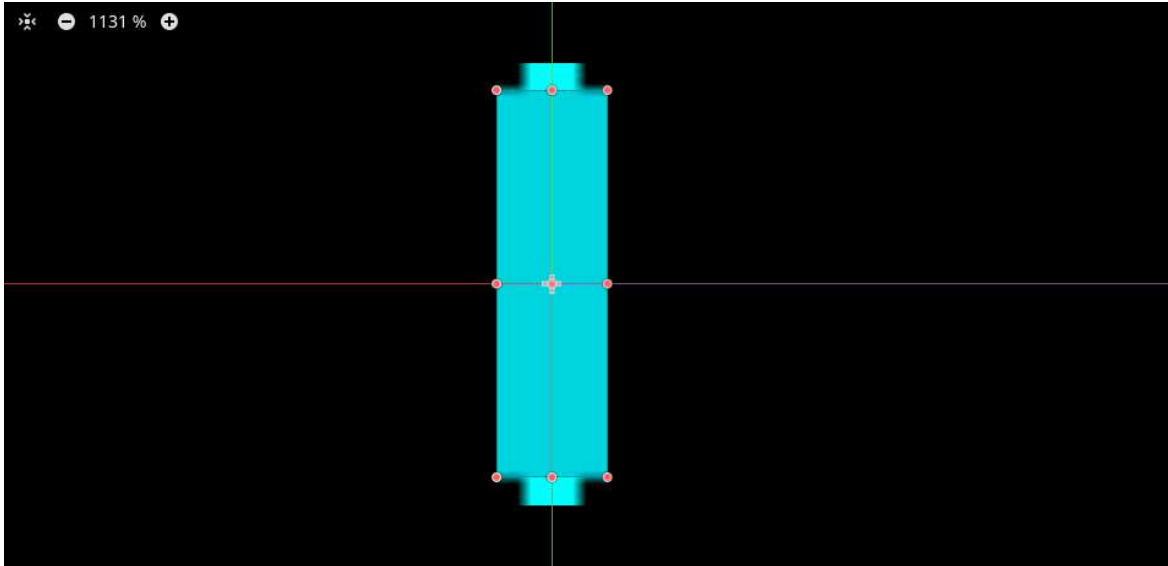
You should now see the image at the point 0,0 - right over the crosshair at the top left of the viewport.



18. Select the Collision2D node and examine the Inspector tab. Select the Shape drop-down menu and choose “New RectangleShape2D”.



Zoom in on the CollisionShape2D in the viewport; it should be overlaid with the Sprite2D node. Drag the boundaries of the CollisionShape2D node to fit the Sprite2D node as demonstrated below.



19. Select the StaticBody2D node and examine the Inspector. Open the Node2D menu and change the scale to 5; the x and y scales are linked by default, so either can be changed.

We have now created the basis from our walls.

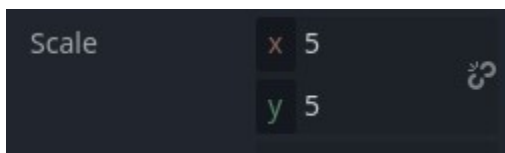


20. Now that we've created a basic Wall scene, we need to now instantiate some instances of this in our main scene to actually add them to our game. Reselect the Main scene and select the StaticBody2D node in the Walls container node. Duplicate this node three times by pressing "ctrl+D" and rename the nodes to "LeftWall", "RightWall", "TopWall", and "BottomWall".



Note that we can't edit the properties of these node's children directly; we would have to make changes within the Wall scene to edit those.

21. Now we'll make direct changes to each of these nodes to create the outline for our level. Select LeftWall and examine the Inspector. Click on the Node2D tab to show the Scale and Position parameters. Unlink the x and y scale by selecting the chain icon to the right of the values.



Set the Scale values to 5,25 and the Position values to 0,324. Your viewport should now look something like this:

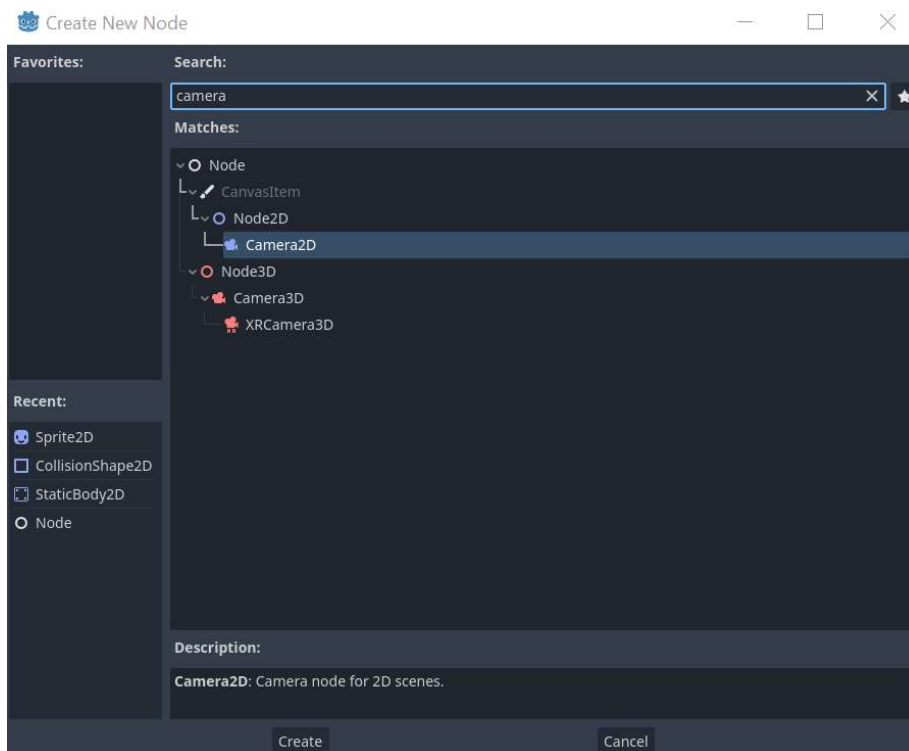


22. Repeat this process for each of the other three nodes with the following values:
  - a. RightWall: Scale (5,25), Position (1152,324)
  - b. TopWall: Scale (5,45), Position (576,0), Rotation (90deg)
  - c. BottomWall: Scale (5,45), Position (576,648), Rotation (90deg)

23. You should now have a box outlined in your viewport as shown below.



24. Now we'll create a camera node to control the camera placement for our level. A camera view is created by default when we make a new project; creating a specific node, however, gives us more control over its placement and is good practice.

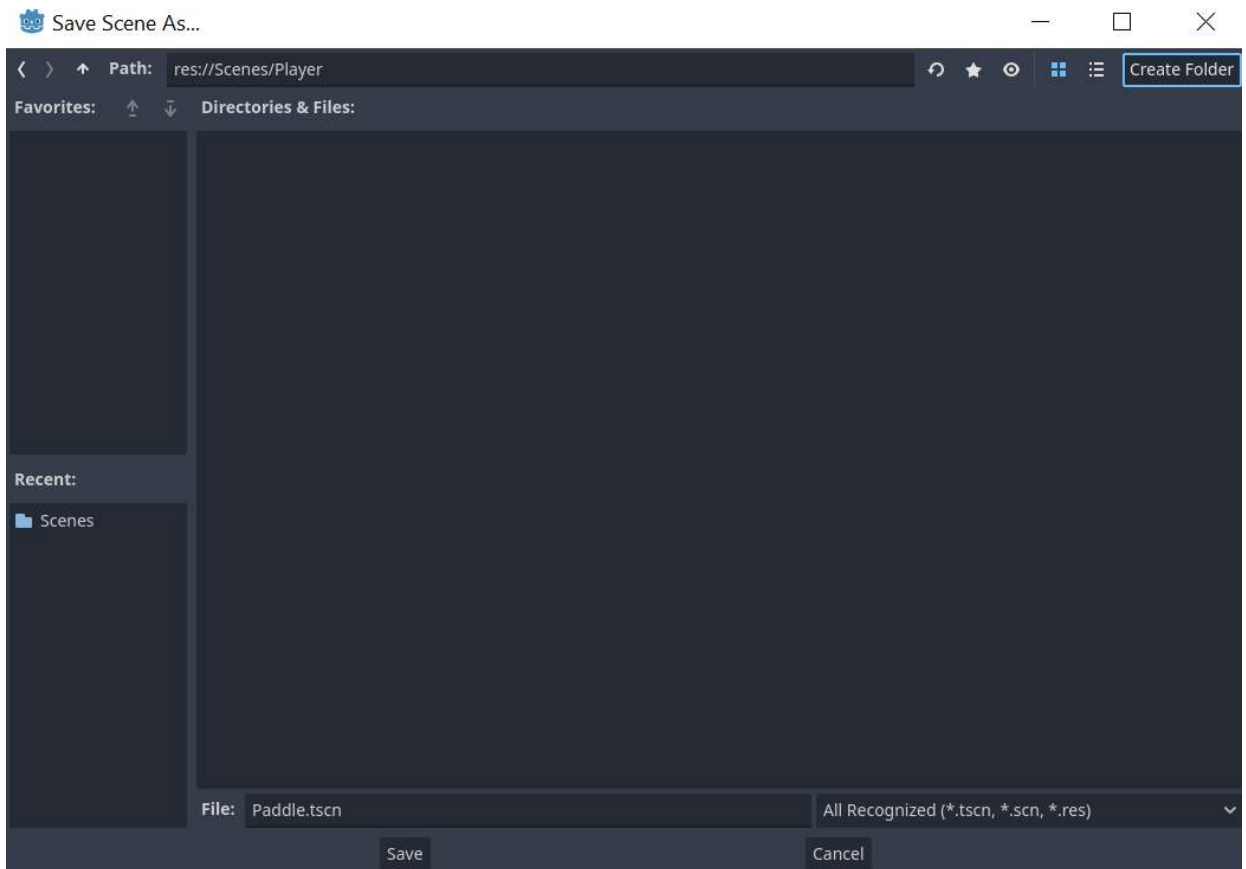


This node must be a child of the Main node such that it is independent of our scenes. Reselect the main scene in the top tab of the viewport and select the main node. Right-click the main node, select “add child node”, and search for “Camera 2D”. Create this type of node.

25. Save your project and press F5 to run it! You should see the same black screen outlined by the walls we inserted in the last few steps.



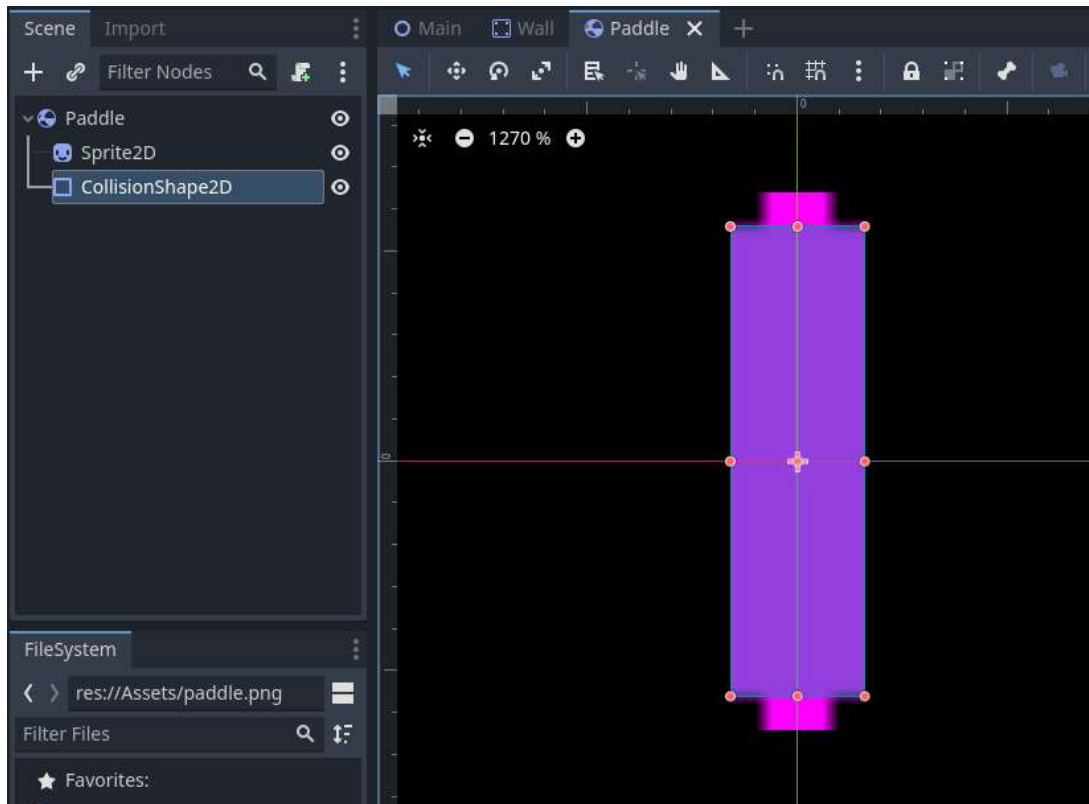
26. Now we want to create a new scene to store our player character; that is, our paddle. To do this, we must create a new node of type RigidBody2D to act as the



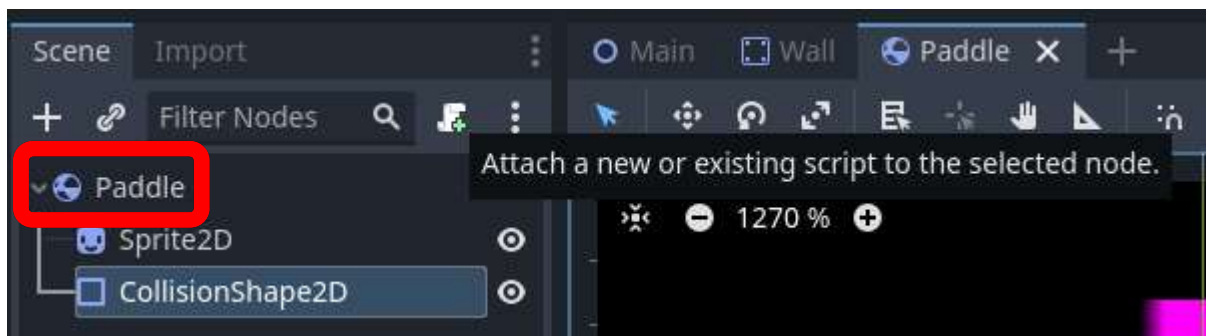
root for our paddle scene. Create this new RigidBody2D-type node as a child of Main and save this node as a new scene with a new folder called "Player" within "Scenes"; name the scene Paddle.tscn, as shown above.



27. In the Paddle scene, rename the root node to “Paddle”. Create another CollisionShape2D node and Sprite2D node. Drag the ‘paddle.png’ texture to the Texture parameter in the Sprite2D node and drag the shape of the CollisionShape2D node to match the Sprite2D node.

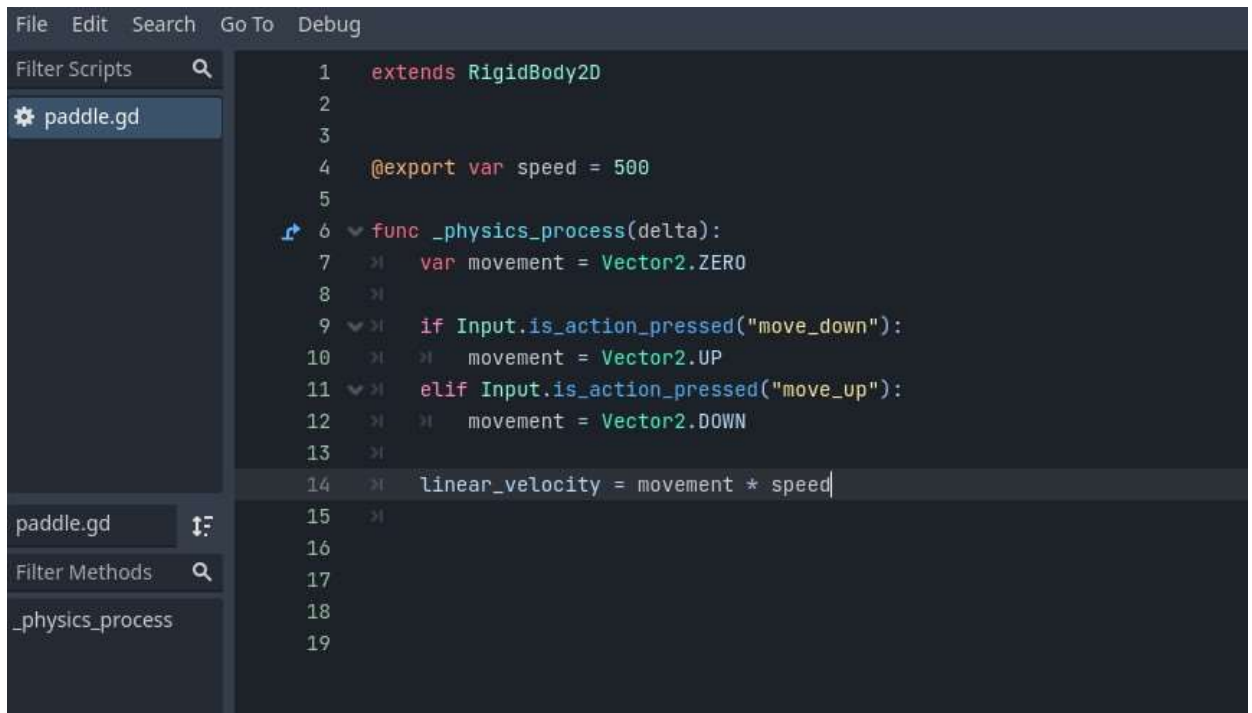


28. Select the root Paddle node. Click the Add GDScript button at the top of the Scene tab to add a script to this scene; this will define our paddle’s movement.



Name the script as ‘paddle.gd’ and save it.

29. Paste the code for 'paddle.gd' from the Game Dev Github into the paddle.gd file.

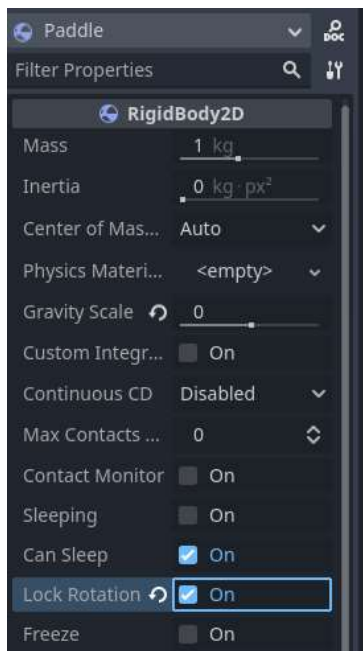


The screenshot shows the Godot 4.0 IDE with the 'paddle.gd' script open. The script is pasted into the 'paddle.gd' file. The code is as follows:

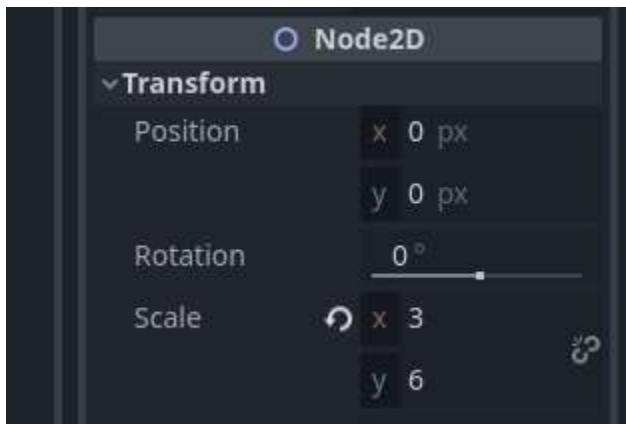
```
1 extends RigidBody2D
2
3
4 @export var speed = 500
5
6 func _physics_process(delta):
7     var movement = Vector2.ZERO
8
9     if Input.is_action_pressed("move_down"):
10         movement = Vector2.UP
11     elif Input.is_action_pressed("move_up"):
12         movement = Vector2.DOWN
13
14     linear_velocity = movement * speed
15
16
17
18
19
```

I encourage you to explore why this code works and how we implemented the input global variables we created earlier in this tutorial.

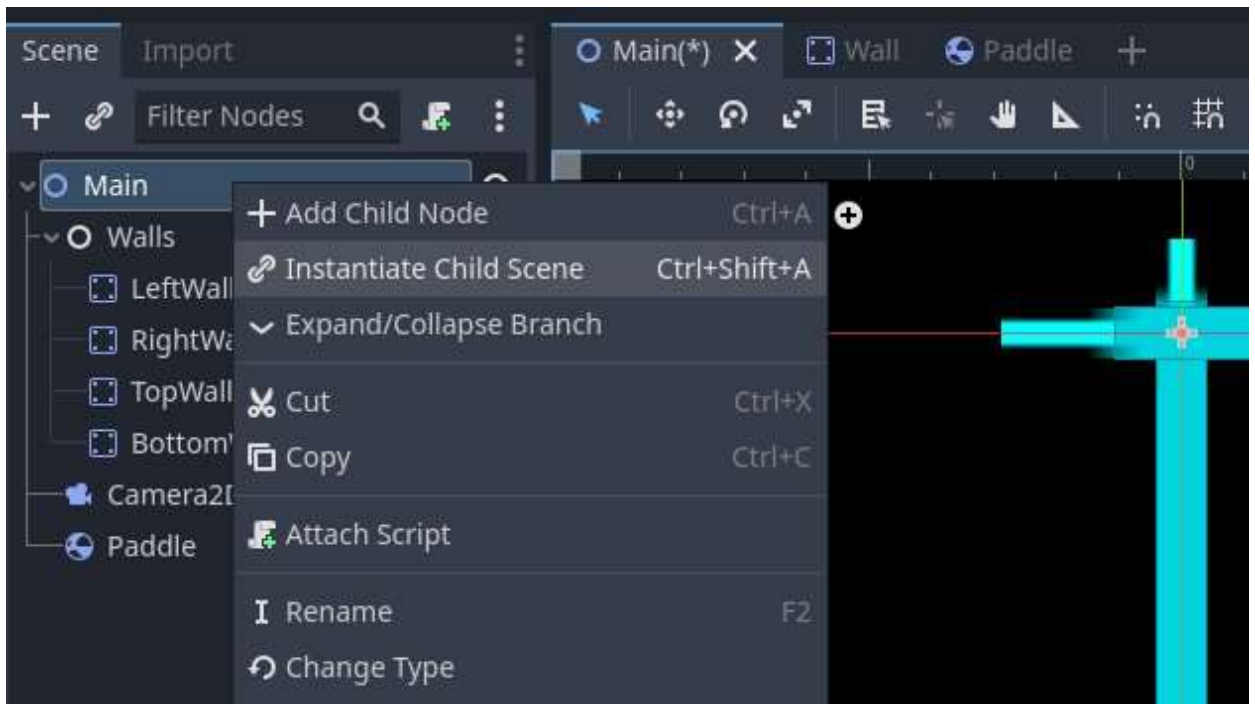
30. Set the Paddle node's gravity scale to 0; otherwise it will be affected by Godot's built-in physics engine. Additionally, the paddle's rotation must be locked, or it will spin every time our ball collides with it.



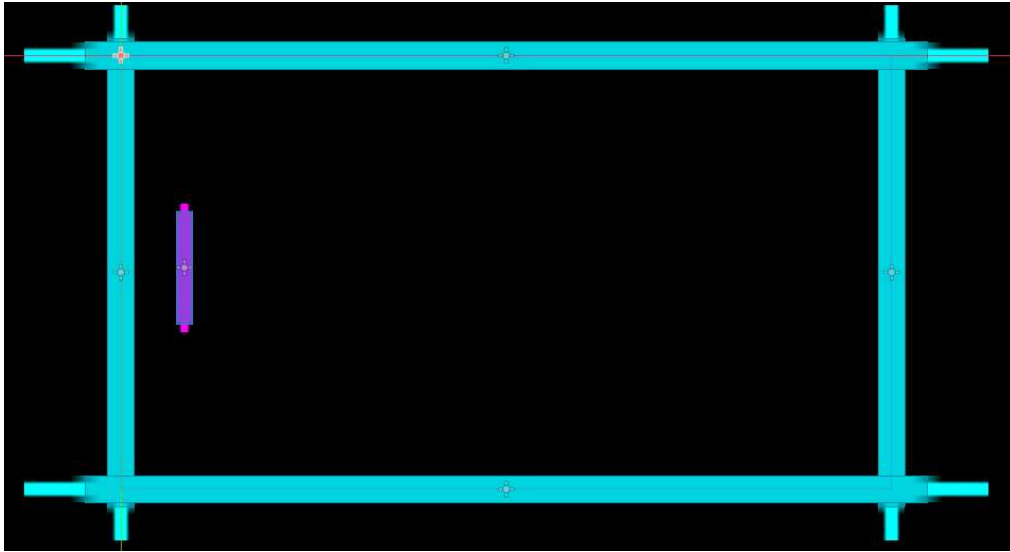
31. Modify the scale settings of both the CollisionShape2D and Sprite2D nodes to (3,6); due to the properties of RigidBody2D nodes, modifying the parent node will result in errors at runtime.



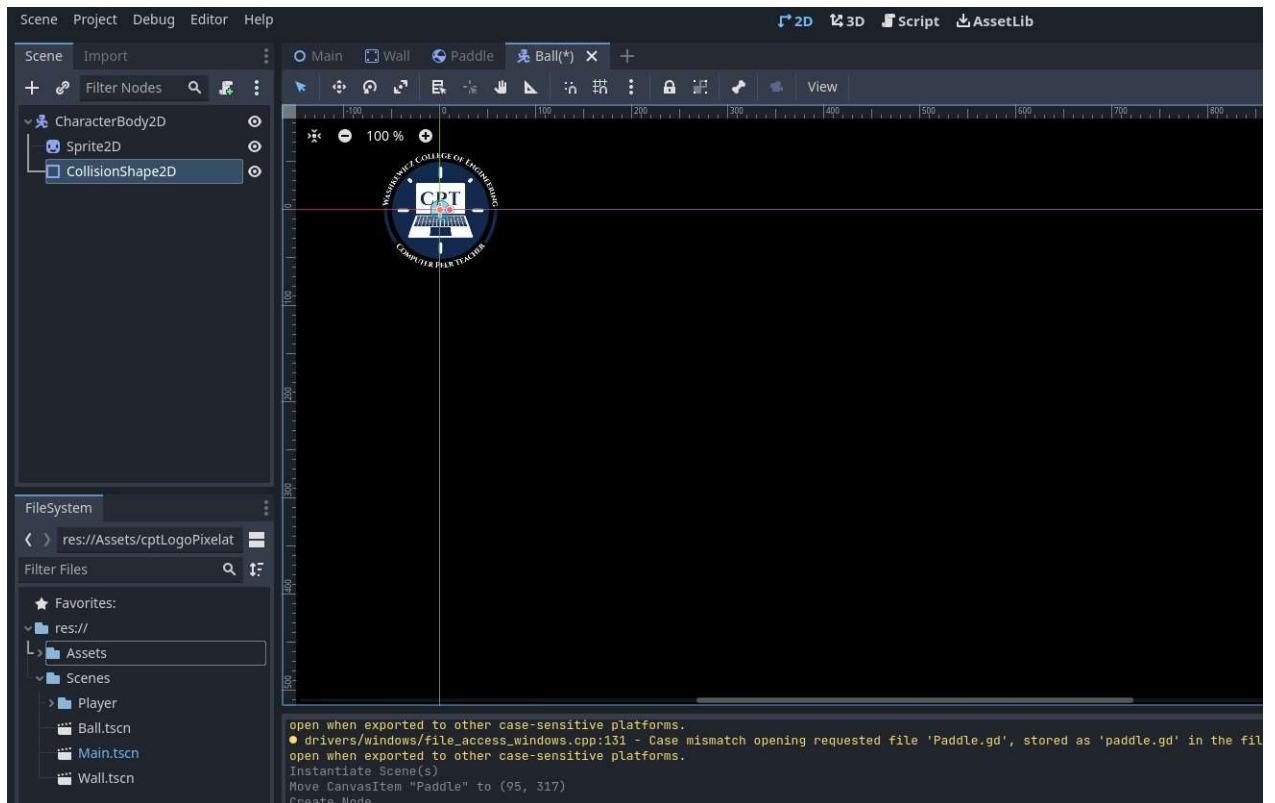
32. As we did with the Wall scenes, we want to instantiate an instance of our Paddle in the main scene. Right click on the Main node in the Main scene and select “Instantiate Child Scene”. Select the Paddle scene within the Player folder.



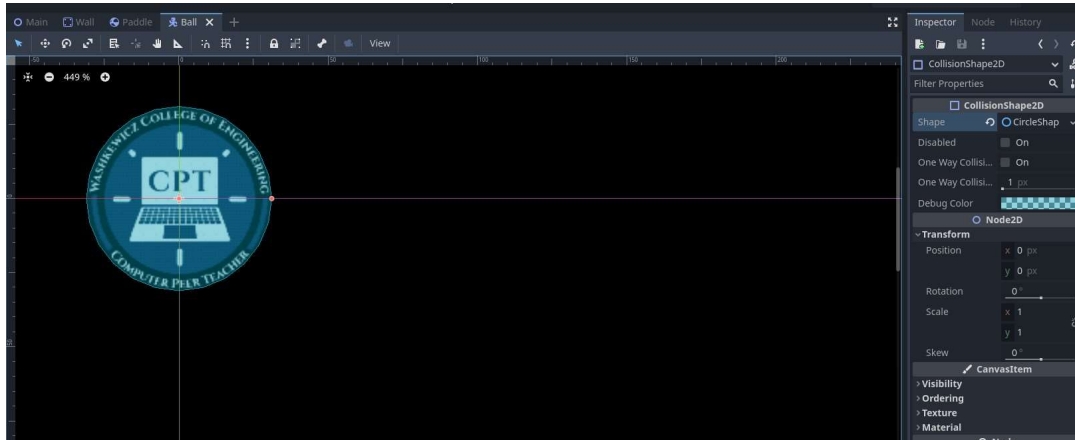
You should now have a paddle in your main scene; if you select the 2D view at the top of the viewport it should be hidden behind the Walls at (0,0). Select the paddle node and drag it where you'd like it to sit in your scene.



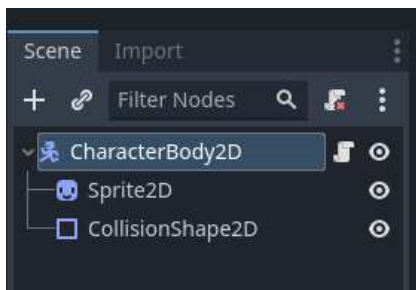
33. Now we have one final element to create: the ball. Create another scene and instantiate the root node as a `CharacterBody2D`. Save this scene within the Scenes folder as "Ball.tscn".



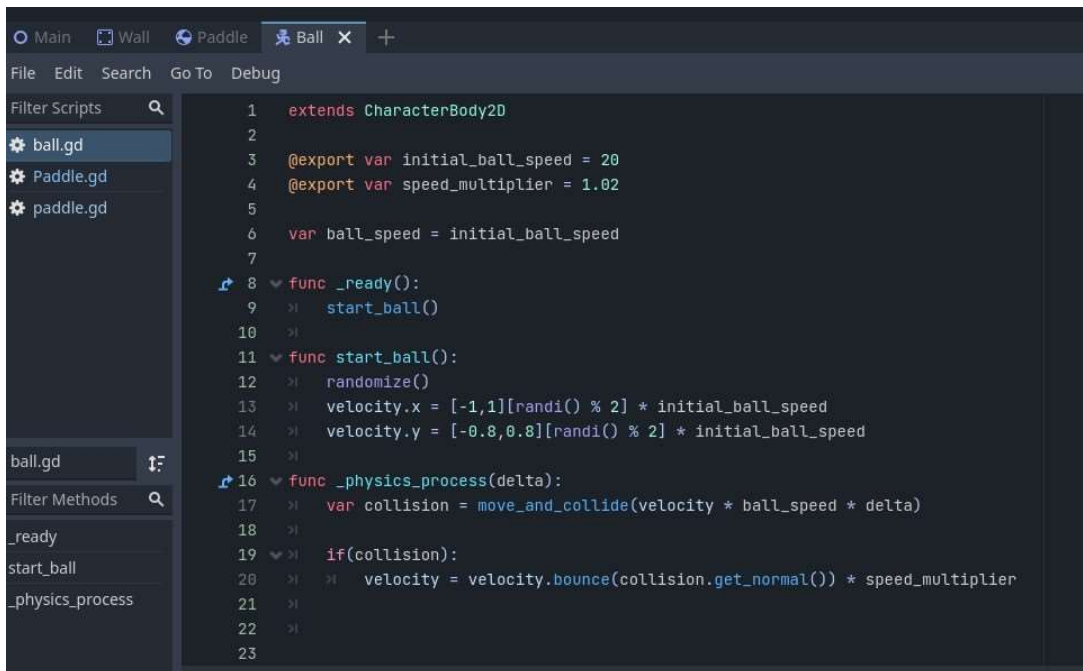
34. Create a `Sprite2D` node and a `CollisionShape2D` node as children. Use the 'cptLogoPixelated.png' as the texture for the `Sprite2D` node and set its scale to (0.5,0.5). For the `CollisionShape2D` node, set the shape to `CircleShape2D` and drag the outline to match the `Sprite2D` node.



35. Create a new GDScript attached to the CharacterBody2D node. Name the script as 'ball.gd' and save it.



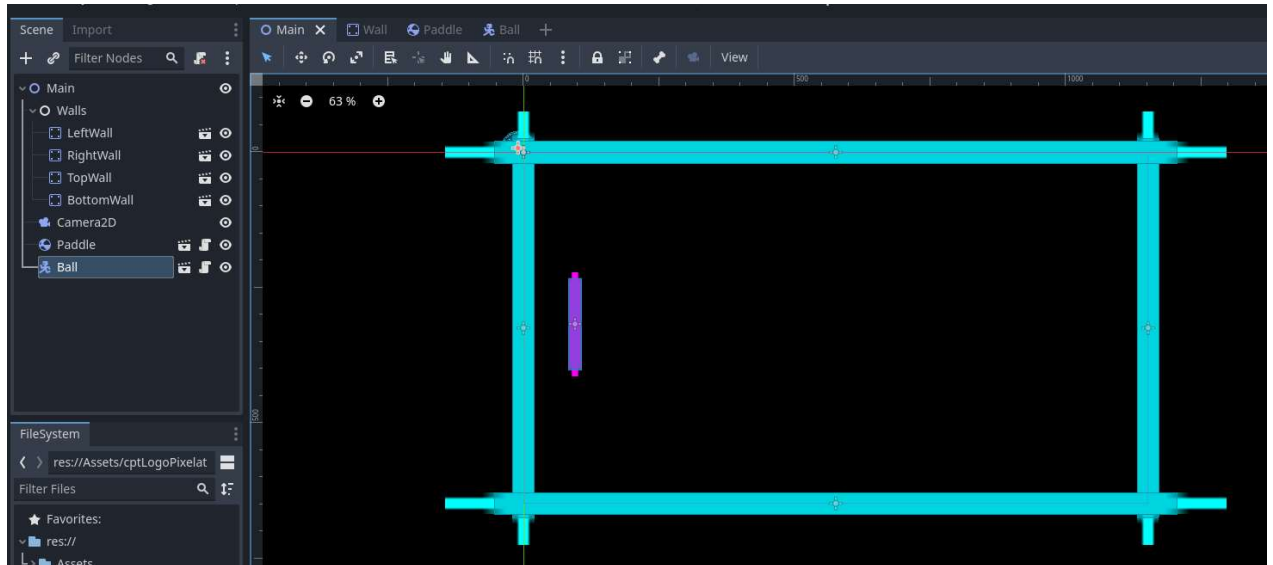
Copy and paste the code from the 'ball.gd' file in the Game Dev Github into the ball.gd file.



Again, I encourage you to dig into why this code works to set the ball's physics and

control its bouncing.

36. Now that we have created a Ball scene, we can instantiate an instance of this in our Main scene. Right click on the Main node in the Main scene and select “Instantiate Child Scene”.



Now you should have a ball created within your main scene viewport. Drag the ball somewhere into your level, save your project, and press F5!

37. Congratulations! You have progressed through this tutorial to create a basic video game with Godot! This game is missing many elements which make up a complete game, but that's the challenge we leave you with today! Use this as your starting point on your

journey into game development!

