

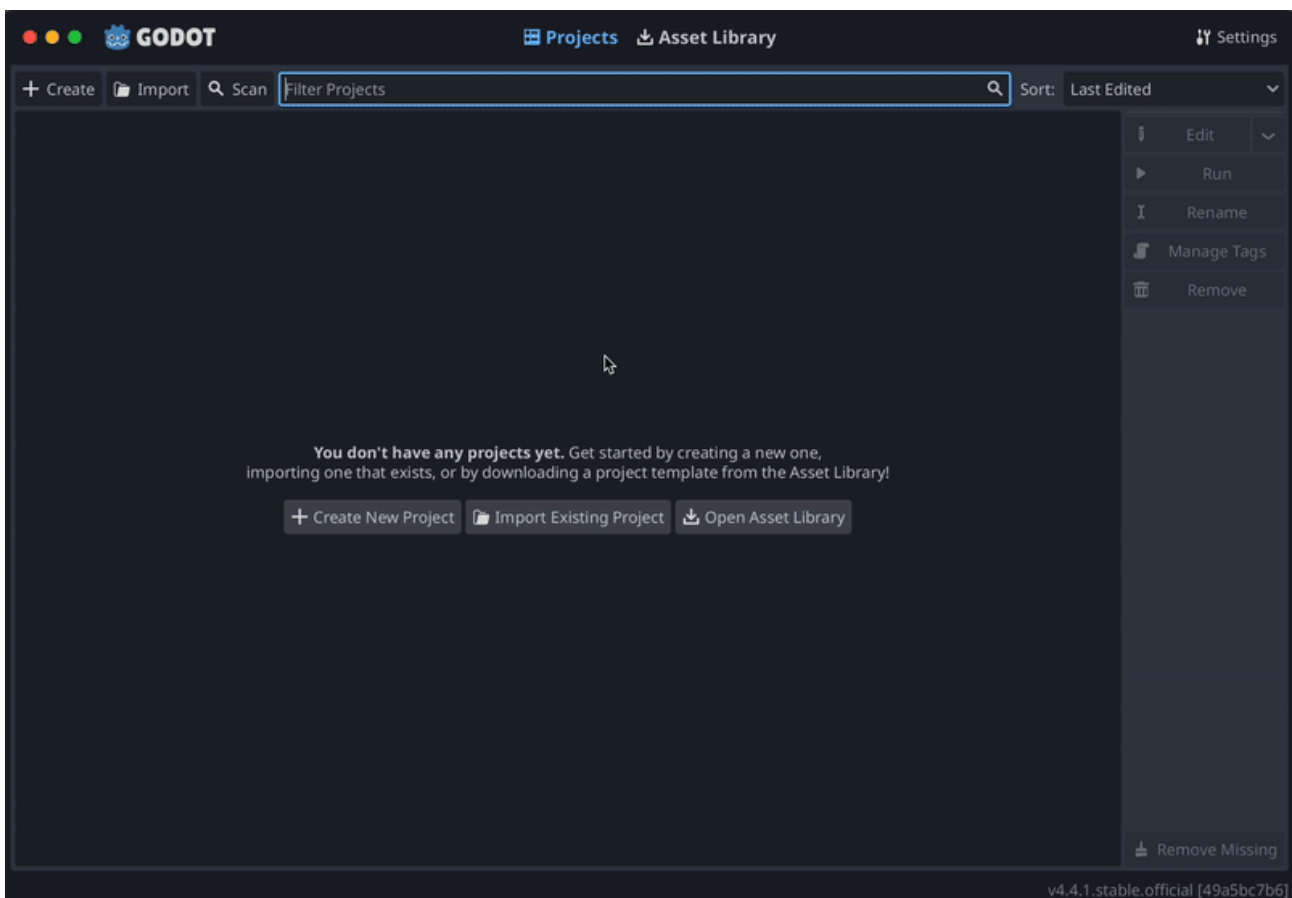
Godot Platformer

In this guide, you'll build a 2D platformer game with a character that can run and jump around. You will use [Godot](https://godotengine.org), a free and open source game engine that is super easy to learn. By the end (~100 mins), you'll have a working platformer you can customize and share!

Setup Godot [🔗](#)

[Godot](https://godotengine.org) is a free, open source game engine. Game engines are used to make games easily. Go to godotengine.org to install it. If you cannot install it, you can also use the web editor at editor.godotengine.org.

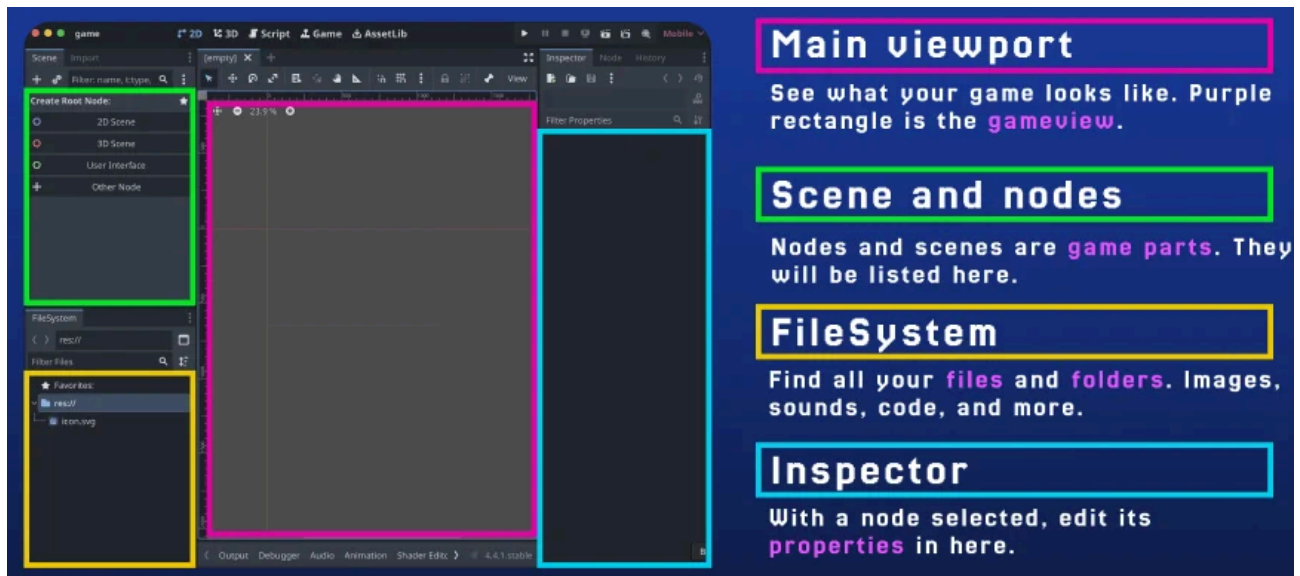
Open Godot, click Create New Project, name your game, select Mobile render, for Project Path select a Game folder, and click Create.



Building the Base of Your Game [🔗](#)

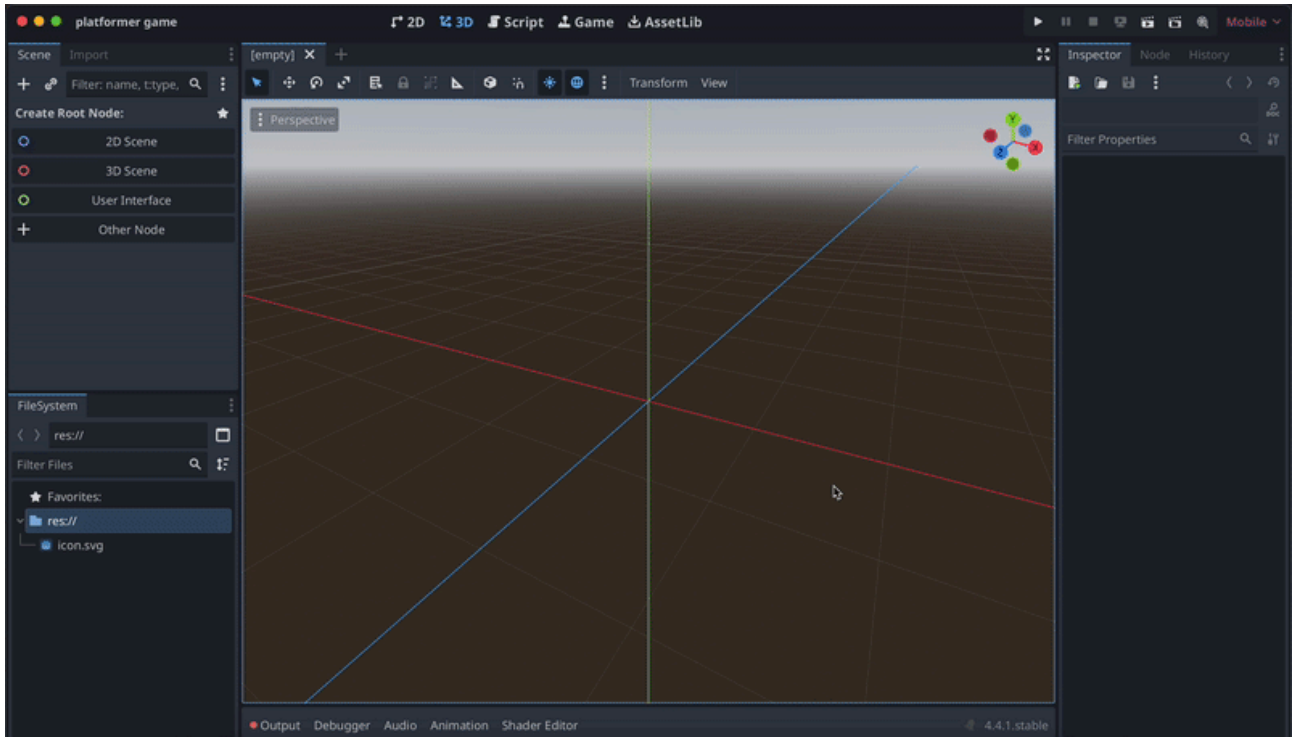
Godot Layout [🔗](#)

When you open Godot, you'll see a bunch of different sections. Take some time to familiarize yourself with the layout of Godot. Also use the top bar to change to 2D view instead of 3D view, since we are making a 2D game.



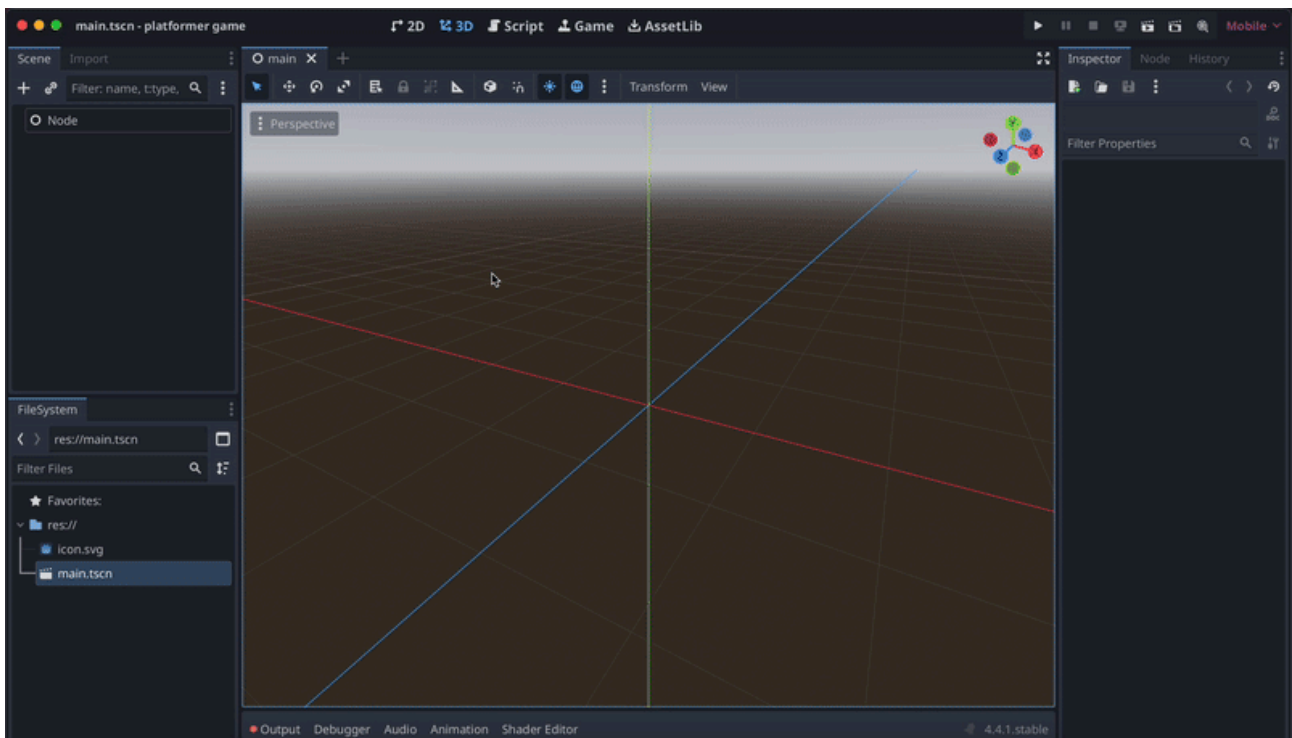
1. Create Scene [🔗](#)

In the Scene section, you'll see options to create a root node, which is the starting point for your game. All the components will be in the main node. Select Other Node and select Node (with the white circle). Save your project with cmd/ctrl+s and title the scene main.tscn. You will see it show up in the FileSystem area below the Scene area.

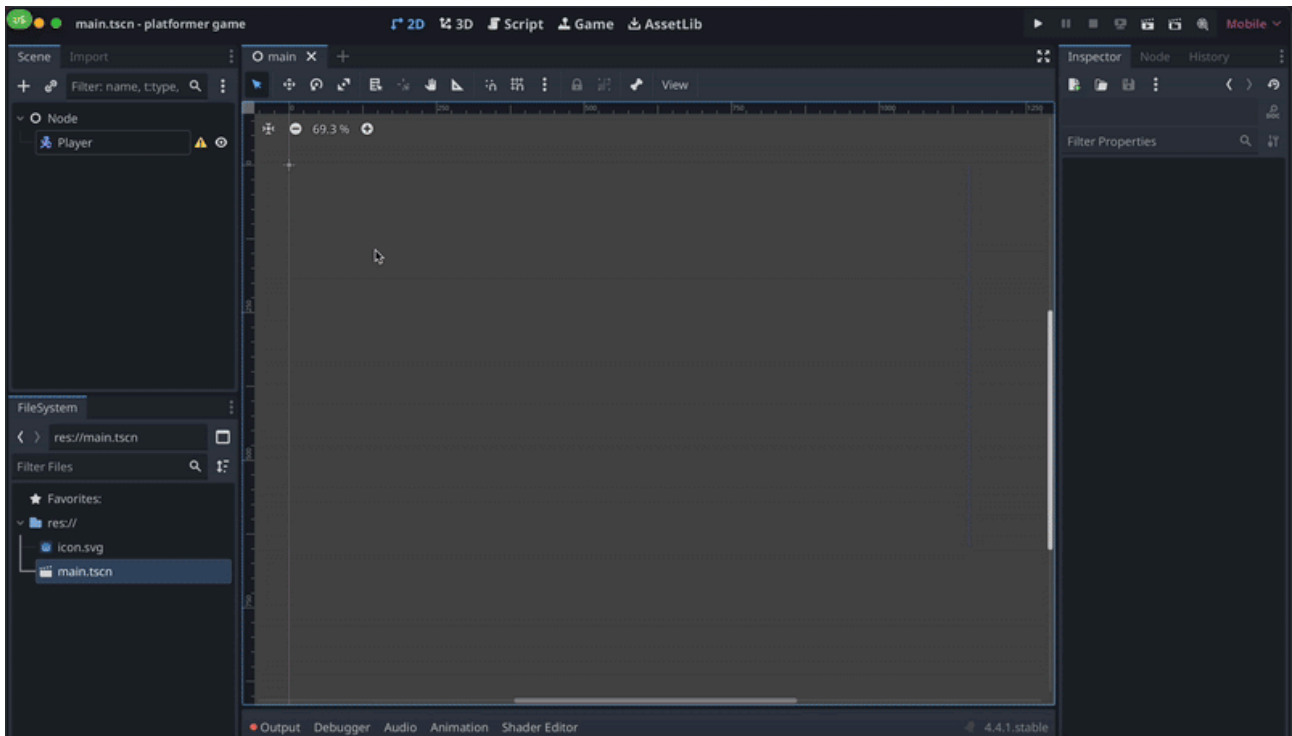


2. Create Player [🔗](#)

Under the Node you just created, use the plus button or cmd/ctrl-A to create another node, `CharacterBody2D`. You'll see many different types of nodes that all have properties that make them fit a different purpose. In the Scene area, rename `CharacterBody2D` to be called `Player`.

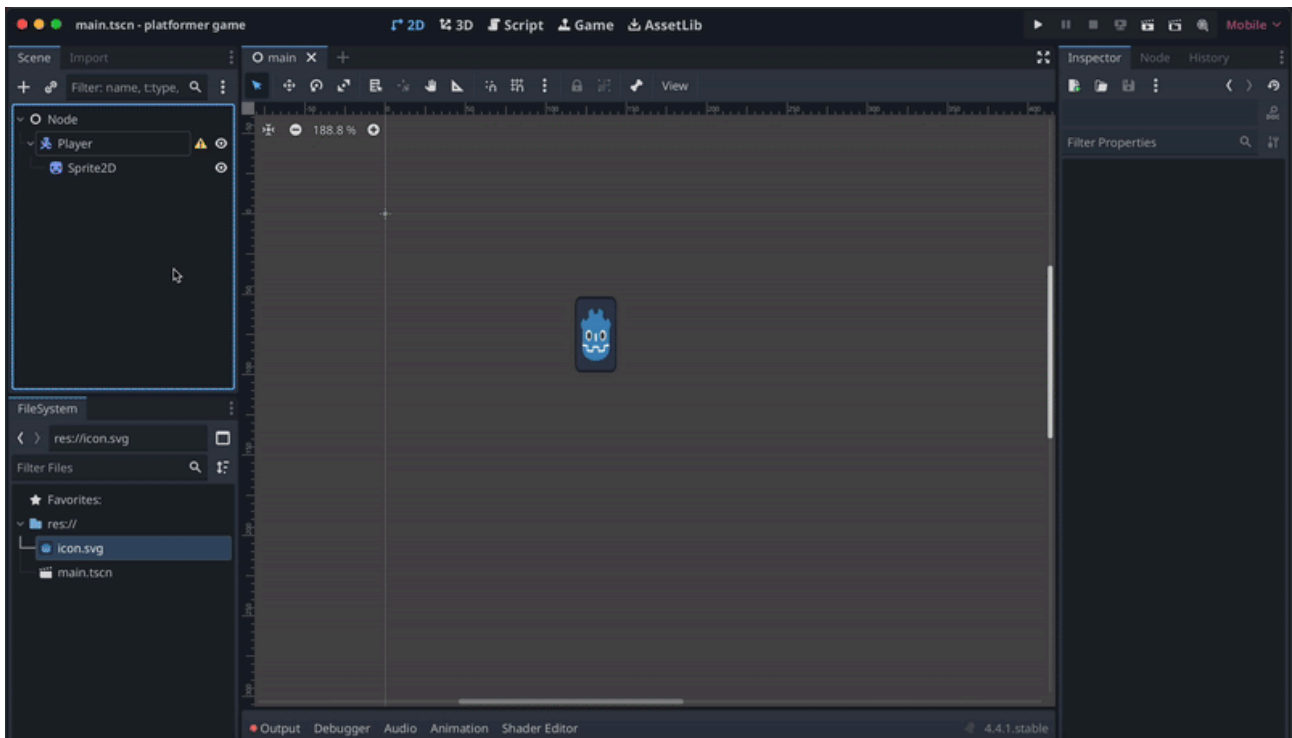


Sprite2D is a node that displays textures (images). Create a Sprite2D node under Player. In the Inspector section on the right, you'll see a Texture box that is currently empty. In the FileSystem, there is an icon.svg of the Godot logo, which will be our placeholder player image for now. Drag that image from the FileSystem into the Texture's box. You can then resize it in the main view. Size it accordingly to the thin purple rectangle in the main view, which is the game view.



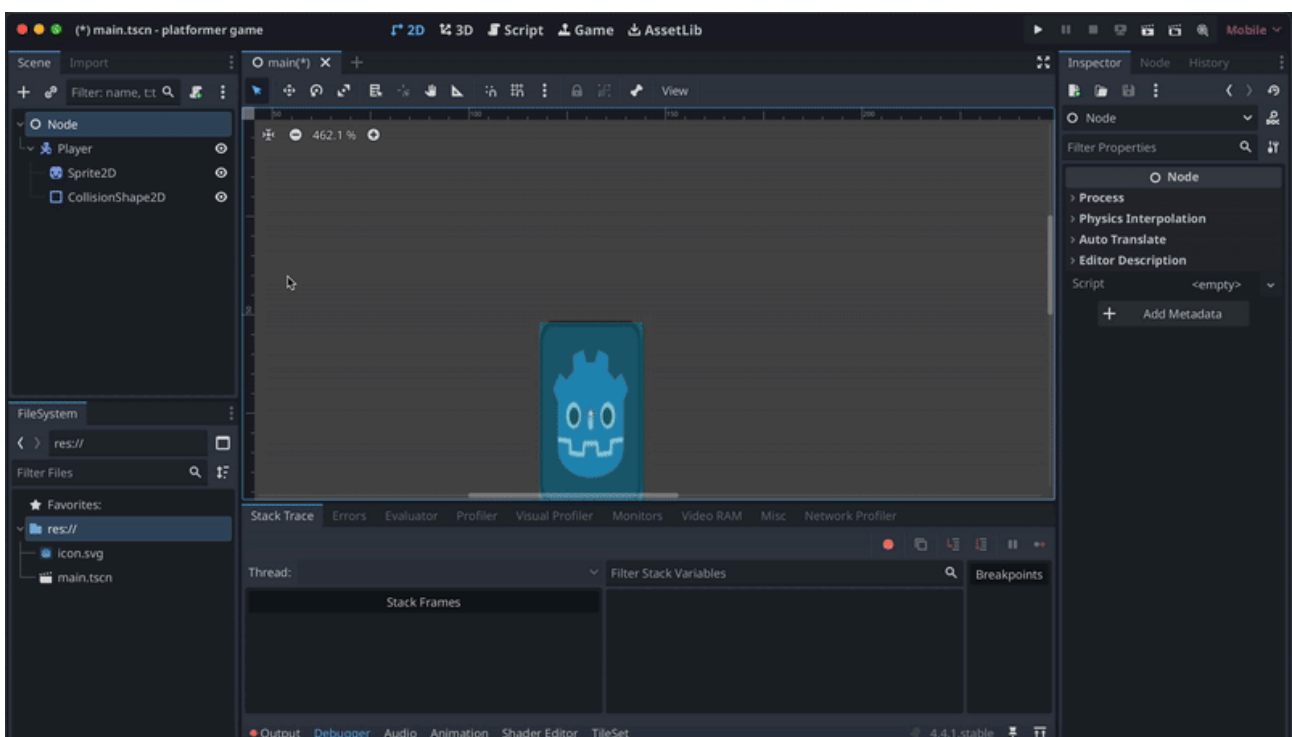
3. Add a Collision Box [🔗](#)

Add a CollisionShape2D to the Player. In the Inspector, select Rectangle as the shape. Resize the blue box that appears in the main view to cover the player. This node helps with the game physics when the player is jumping and walking around, so Godot will know the bounds of the player.



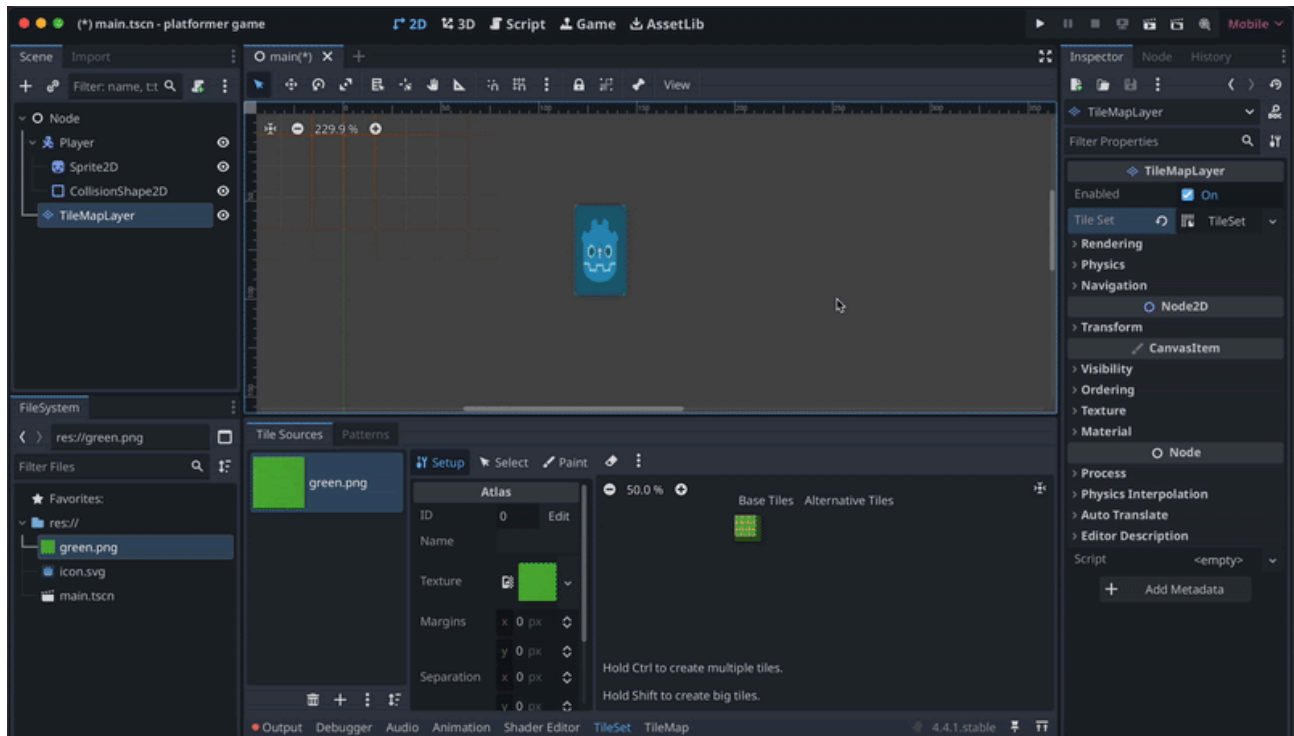
4. Create a Tileset [🔗](#)

Tilesets are used to easily make and customize grid-based 2D maps in games. Add a `TileMapLayer` node under the main Node (not under Player). In the Inspector, select `New Tileset` in the `Tile Set` option. In the bottom bar, navigate from `TileMap` to `Tileset`. Find an image for your tileset, a plain color will do for now. Add it to your `FileSystem` by simply dragging it in. Then, drag that image from the `FileSystem` into the `TileSet` area. When prompted, select `Yes` to modifications.



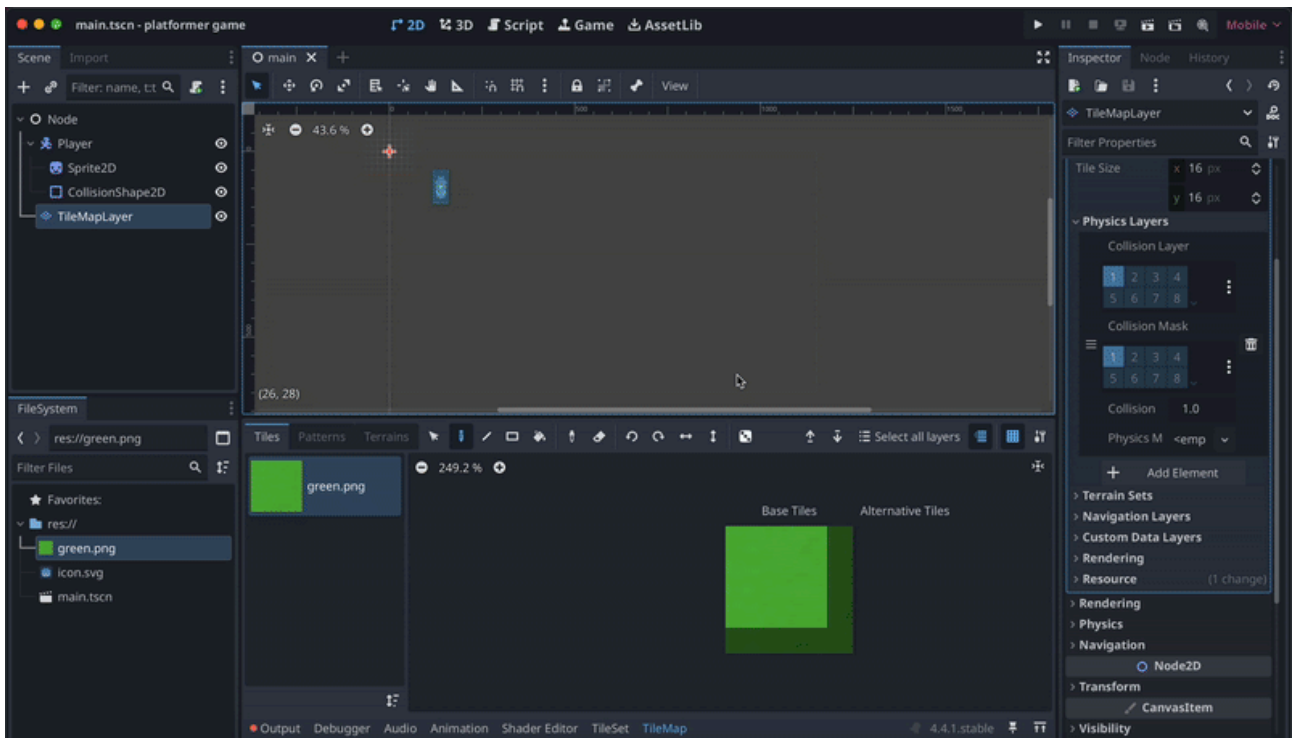
5. Add Collision For Tileset [🔗](#)

Just like the player, each block of the map needs a collision box too (or else the player would just fall through the ground). Open TileSet in the Inspector and under Physics Layers, click Add Element. Then in the TileSet area in the bottom, change to the Paint tool and select Physics Layer 0 as the Paint Property. Then, select all the squares.



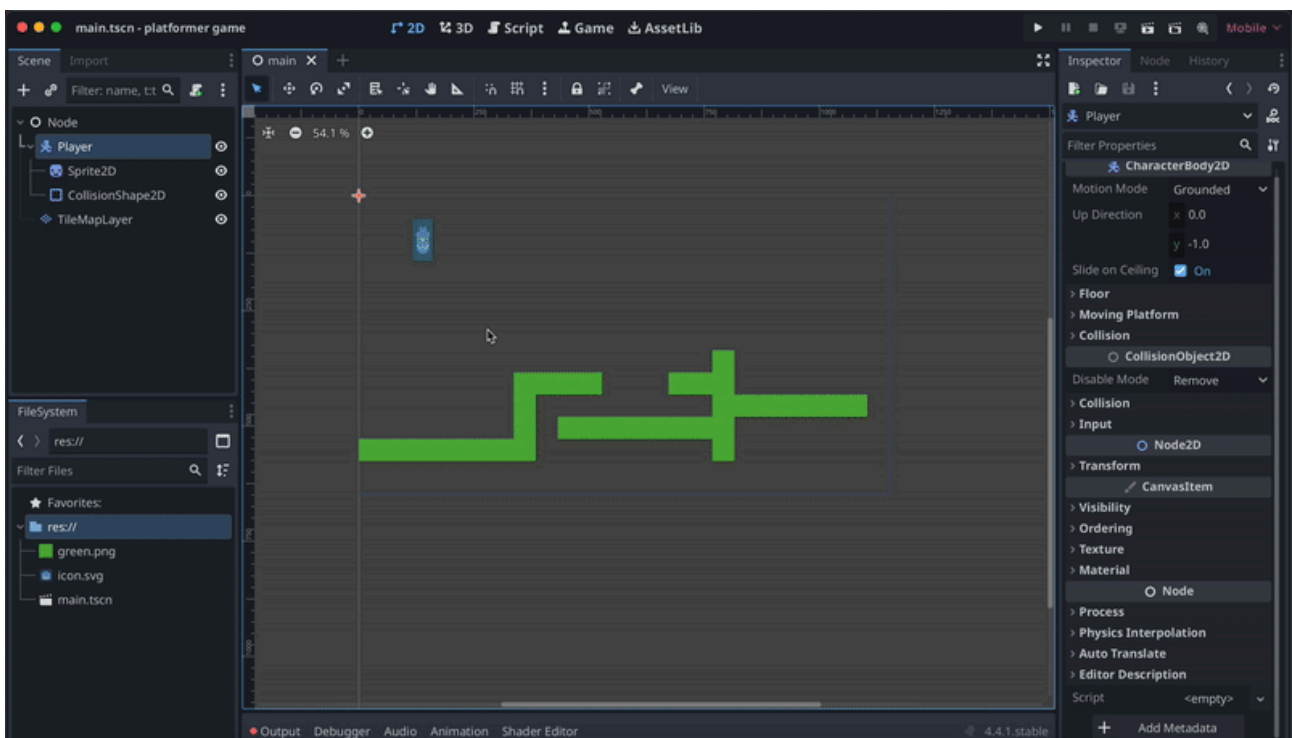
6. Draw a Map [🔗](#)

In the bottom area, change from TileSet to TileMap. Use the pencil tool and select one cube. Then, you can freely draw a map in the purple rectangle (game view) in the main view. Design your game map however you want! You can also adjust the block size in Inspector, where there is Transform and Scale.



7. Create a Player Script [🔗](#)

To make it so the player can react to key presses by jumping and walking around, you need to code a script for the Player. Select the player, click the scroll with a green plus (in the Scene area) to create a script. Choose GDScript (the coding language Godot uses), check the Template box and choose `CharacterBody2D: Basic Movement`, then create the script.



8. Understand the Template Code

You can run your game now with ctrl/cmd+b or the play button. It works, but the template code is the minimum and we can improve it to make the movement better. First, it is important to understand the template code! Read through:



```
extends CharacterBody2D

const SPEED = 300.0
const JUMP_VELOCITY = -400.0

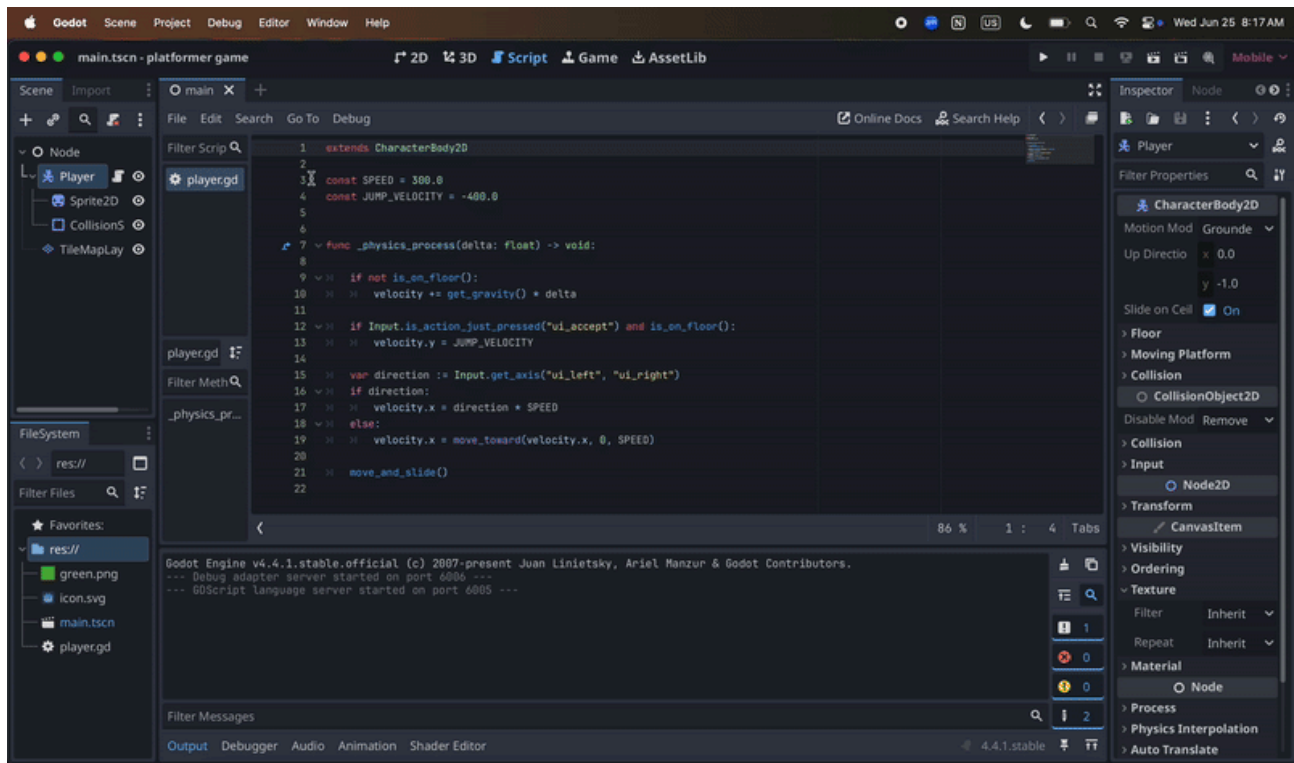
func _physics_process(delta: float) -> void:
    if not is_on_floor():
        velocity += get_gravity() * delta
    if Input.is_action_just_pressed("ui_accept") and is_on_floor():
        velocity.y = JUMP_VELOCITY
    var direction := Input.get_axis("ui_left", "ui_right")
    if direction:
        velocity.x = direction * SPEED
    else:
        velocity.x = move_toward(velocity.x, 0, SPEED)
    move_and_slide()
```

Annotations:

- `extends CharacterBody2D`: this tells Godot that this script is connected to the **CharacterBody2D** node
- `const`: `const` means the value doesn't change in the game. **SPEED** is how fast the player goes, **JUMP_VELOCITY** is the jump height. In Godot, negative numbers go up and 0 is the top of the screen.
- `_physics_process(delta: float)`: the **physics_process** function runs many many times every second and **delta** is the time since the last frame, it helps keep the animation smooth
- `if not is_on_floor():`: if the player is **not on the floor**, then they are in the air and should fall because of gravity.
- `velocity += get_gravity() * delta`: **get_gravity** is the gravity strength, and **velocity** makes it so the player falls faster over time
- `if Input.is_action_just_pressed("ui_accept") and is_on_floor():`: **ui_accept** is space or enter and if the player is **on the floor**, then sets **velocity** to go up, so the player can jump
- `Input.get_axis("ui_left", "ui_right")`: **Input.get_axis** returns -1 when the **left** key is pressed and 1 when the **right** key is pressed. **velocity.x** makes the player move in the correct **direction** at **speed**. If they are not pressing right or left, then the **velocity** is slowly moved towards **0** to a stop.
- `move_and_slide()`: this is the final important function which applies all of this to the player!!

9. Make an Input Map

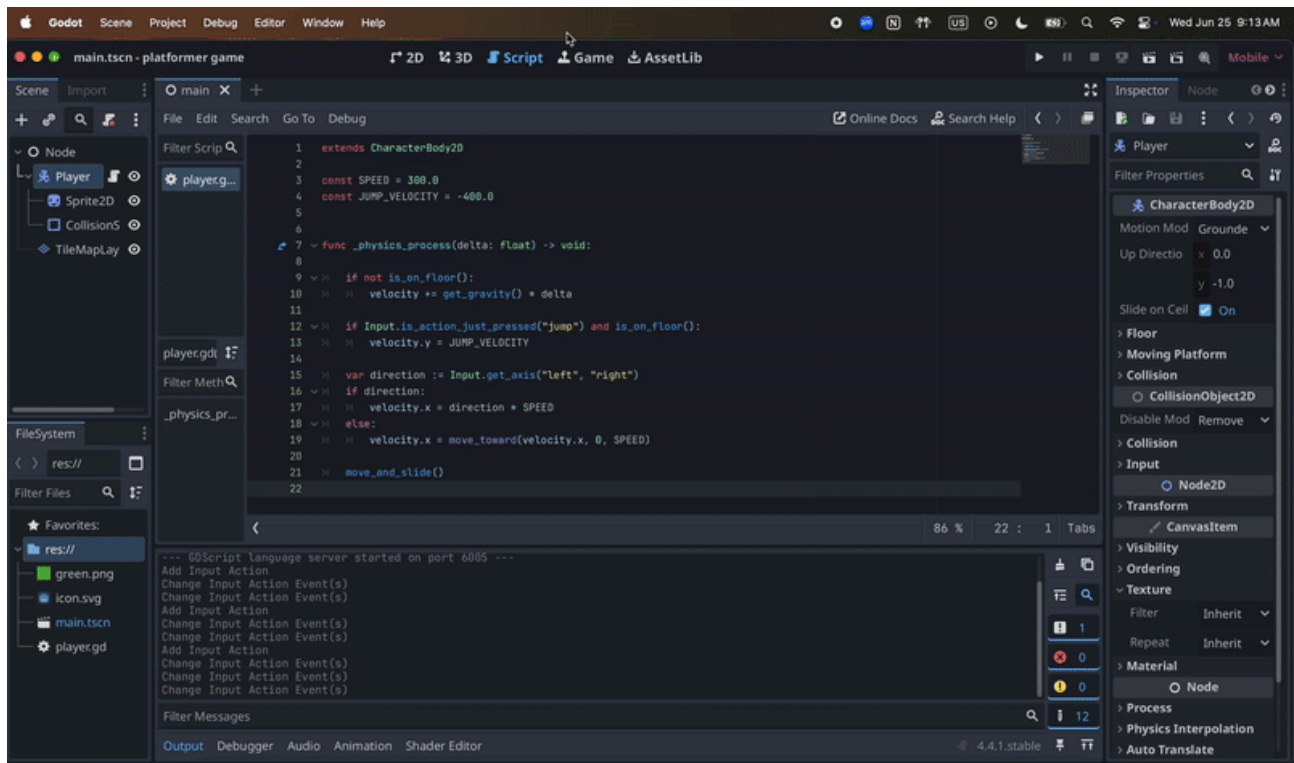
In the template code, the keys to move are `ui_accept`, `left`, and `right`. The user can't even use the up arrow to jump! We can adjust these and also add WASD controls using an input map. At the very top, go to Project, Project Settings, then Input Map. Add three new actions called `right`, `left`, and `jump`. In each action, use the plus button on the right to add a key, which you can just press (WASD, arrows, space).



In the player.gd script, replace “ui_accept”, “ui_right”, and “ui_left” with “jump”, “right”, and “left”.

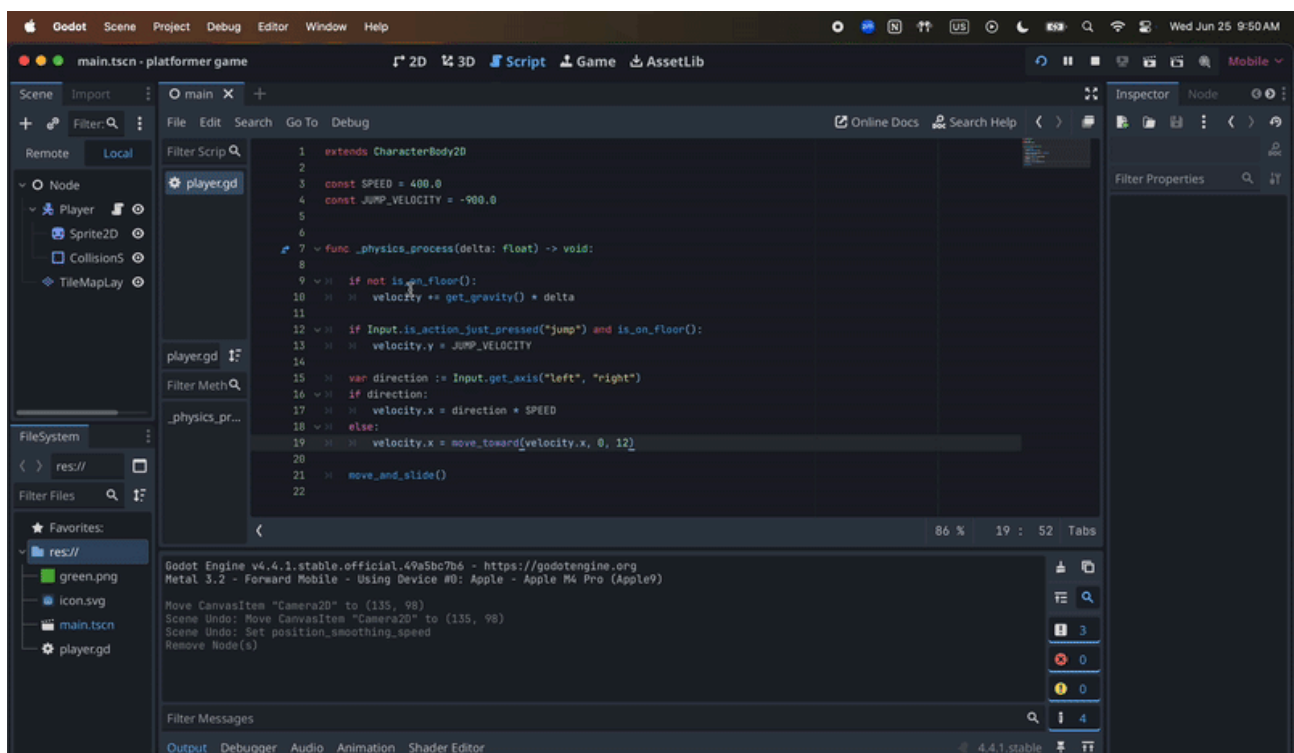
10. Tweak Movement in Code [↗](#)

The current code works, but the player’s movement is rather slow. We can make some small changes in the code to make the movement sharper. On line 19, you can change the last value (SPEED) to other numbers, such as 900, to control how fast the player comes to a stop. You can make the gravity higher via Project, Project Settings, General, 2D Gravity, to a number like 2500. Change JUMP_VELOCITY to -900, making the Player jump higher. Use ctrl/cmd+b to run your game and test.



11. Add a Camera [🔗](#)

To make the camera follow the Player as it walks through the game, add a Camera2D node under Player. Edit its position in the main view so it is centered on the player. You can also add a cool camera lag effect in Inspector, enabling Position Smoothing, and setting the speed to be around 7. The lower the speed, the more camera lag there.



12. Congrats

The base of your platformer game is finished! You can run the game and see.

Customize Your Game

Now that you have all the base pieces, it's time to customize your game! Pick a fun theme for your game. Who is your player? Where are they? Where are they going? What does their world look like?

Change Player, Background, and Tileset

You can draw your own player, background, and tileset image, or also find assets online.

Add More Features

Some ideas include:

- **Collectibles** - Create coins, gems, or power-ups that the player can collect. Use Area2D nodes and signals to detect when the player touches them. [This](#) is a useful video.
- **Enemies** - Add moving enemies that the player must avoid or defeat. Check out [this](#) tutorial.
- **Score System** - Track the player's score based on collectibles, time, or other achievements. Display it on the UI using Label nodes.
- **Sound** - Import audio files and use AudioStreamPlayer nodes to add background music and sound effects for things like jumping. See [this](#) helpful tutorial.
- **Anything else you can think of** - There are tons of great tutorials online to help you! Do some searching :). Make your game like no other.

Publish on Itch.io

Once you're done building your game, it's time to upload it online so other people can play it! [Itch.io](https://itch.io) is a popular platform where games are shared.

Exporting from Godot

In the top bar, go to Projects, Export, Add, Web. Then, Manage Export Templates, Download and Install. When it is done installing, go back to Export, select the Web preset and click Export Project. Make a folder called Exports and save the index.html to it. In your File Explorer, compress all the contents in the Export folder into a .zip. Name it your-game-web.zip.

Publishing

Go to itch.io and make an account. In Itch.io's dashboard, click Create New Project. Title it, add a description. For Kind of Project, choose HTML. Upload the .zip file you made. Check the "This file will be played in the browser" box. Change Viewport Dimensions to the ones in Godot that you can find by going to Project, Project Settings, and Window.

Check the Fullscreen button and the SharedArrayBuffer support box. Add tags, a cover image, and anything else you want. You can also edit the theme (colors and font) on your game page.

Caution

You need to Save the page before making it Public. It is currently a Draft, so click Save & view page, then go back, and select Public under Visibility and Access. Make sure your itch.io page is public!

Finally, you can share your game page's URL for others to play. If you make updates in Godot, you just need to reupload a new .zip.

This guide was originally written by Estella Gu for [Jumpstart](#).