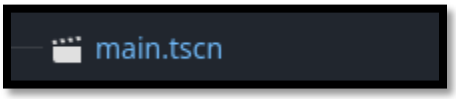
CODE NINJAS®

Bronze Belt Ninja Guide

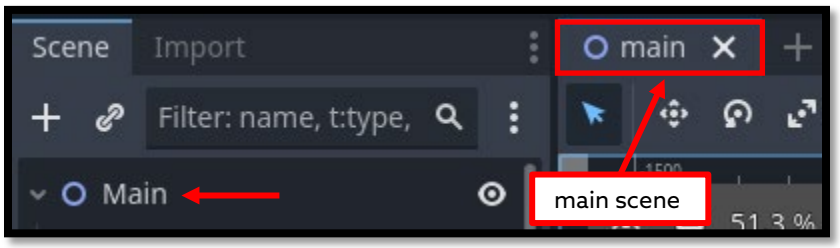
Activity 02: Scavenger Hunt

Scenes

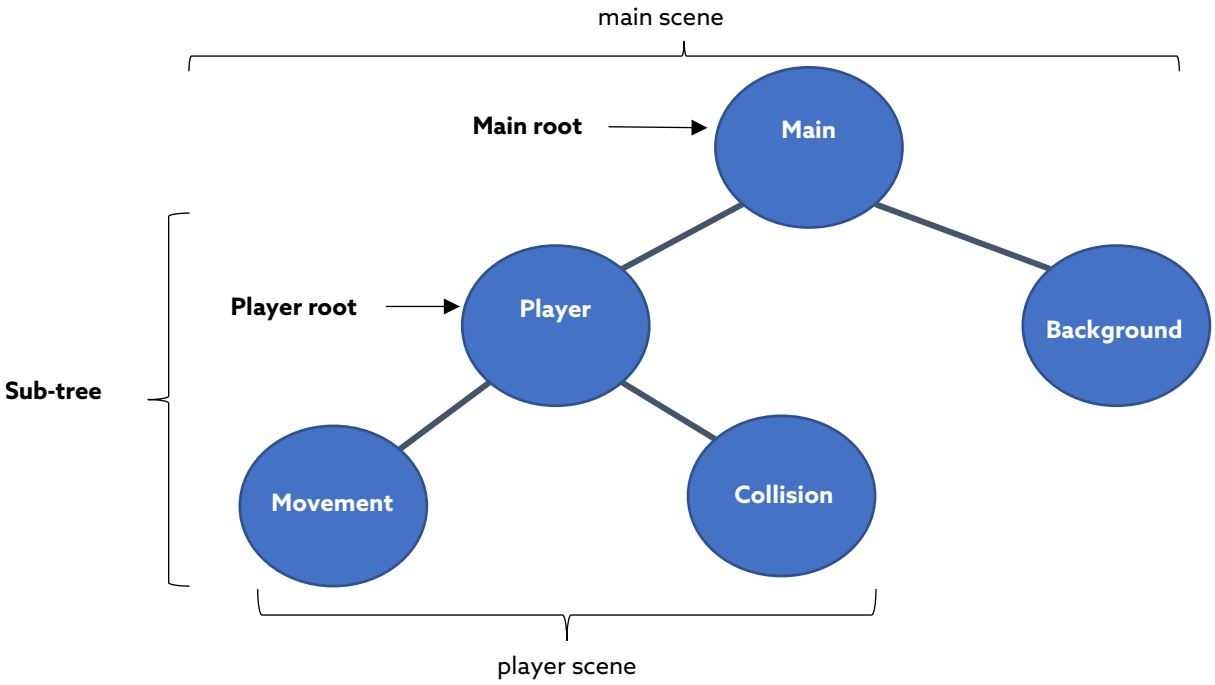
A tree of nodes can be saved into a **.tscn** file called a **scene.** Once a scene is saved, it acts like a blueprint where it can be reused many times throughout a game. Every project always has a **main scene,** which tells Godot where to start the game.



main scene

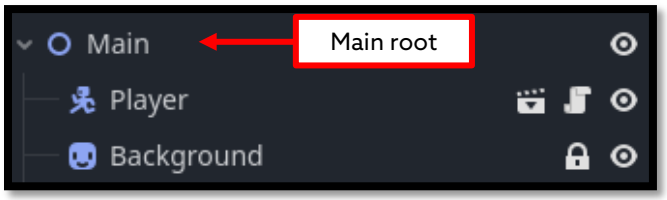


Scenes are not limited to only the **main scene**; they can also be created from any **child node** of the **Main root.** Remember, child nodes are like the branches of the tree. When saving a **scene** from one of these child nodes, it creates a **sub-tree**—a smaller section of the tree with its own root and children.



Sub-trees can contain a single node or multiple child nodes. Remember, the **Main root** is the first node created in the project, but **any node** can be a **root** of its own sub-tree when saved as a separate **scene**.

A **root** is always the node from which the scene was created. For example, in the **main scene**, the root is **Main**; while in the **player scene,** the root is **Player.**



main scene

player scene

Activity 02: Scavenger Hunt

In this activity, you will create a simple platform game with collectables and a score! This project will help you learn about scenes, user interfaces, and platforms.

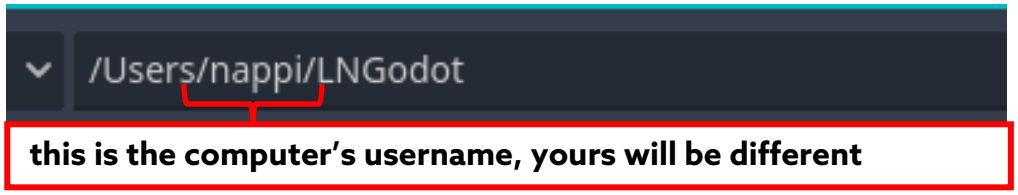
By the end of this activity, you will have explored how to create one-way collisions, importing and navigating scenes, creating a user interface, and customizing a player sprite.

1

All projects will be stored in a path like:

/Users/[MyComputerUsername]/[MyInitials]Godot

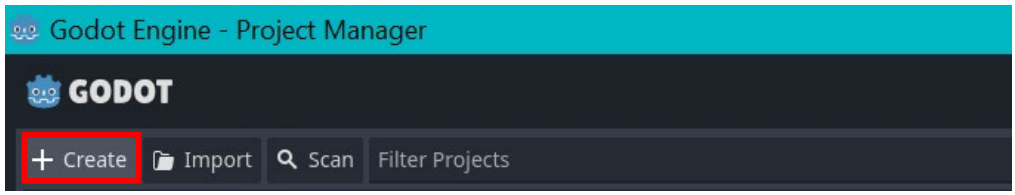
Don't worry if your path looks slightly different from the image shown! All computers have their own username.



2

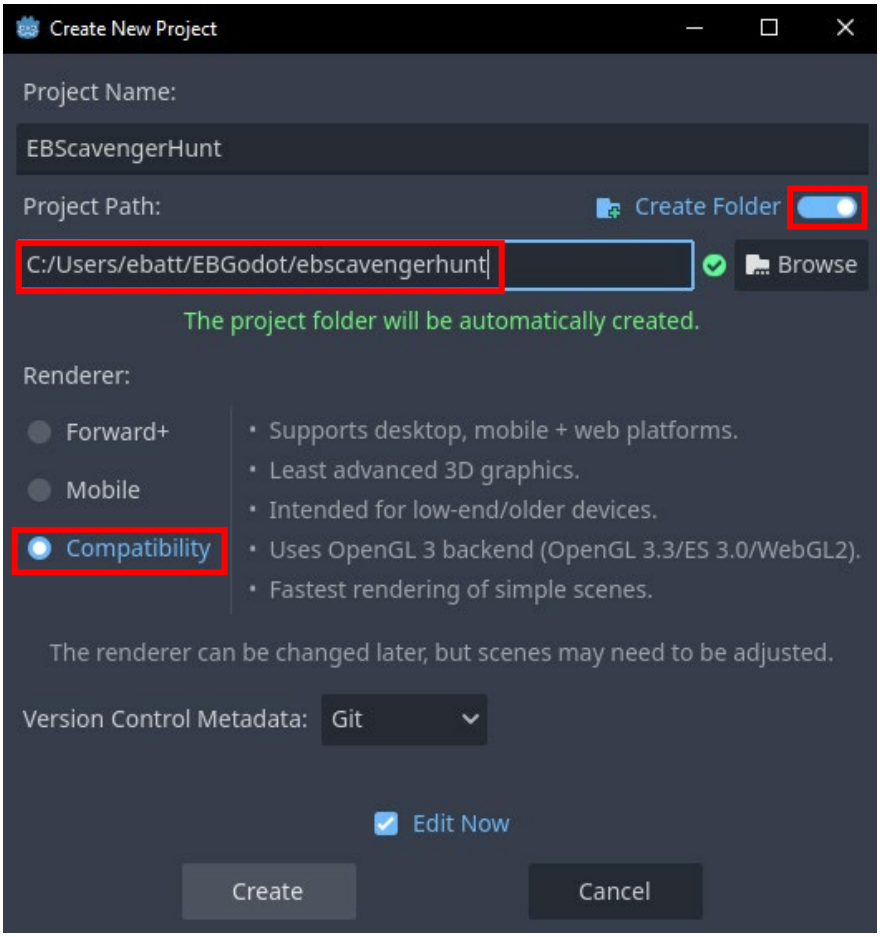
After opening Godot, in the top left corner select **+ Create.**

A **Create New Project** window will pop up. Name the project **MyInitialsScavengerHunt.**



3 Make sure the **Project Path** is the same as specified from a Code Sensei in **Activity 00.** Refer back to **Step 1** for what the **Project Path** might look like.

Check that **Create Folder** is turned on, and that the **Compatibility** mode for the renderer is being used.



Pause for **Sensei Stop #1**!

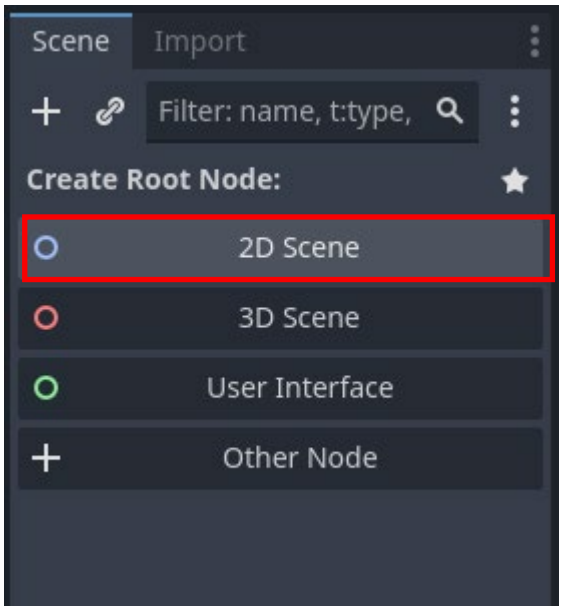
Check in with a Code Sensei before moving on. Make sure the **Project Path** is correct.

4

Click **Create**.

First, the **Main root node** and **main scene** need to be created. This project will be in a **2D environment** so all nodes will be **2D**, instead of **3D**.

In the top left corner, find the **Scene** menu and select **2D Scene**. This will create a **Node2D** and a **2D scene.**

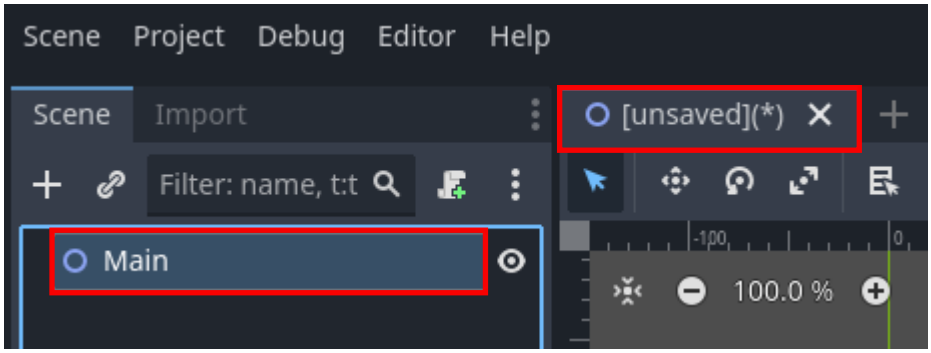


New Concept: Root Node

The Main root node is the first node in a project and acts like the root of a tree. This is the parent to all other nodes in the project. A tree of nodes can be saved to a file called a **scene**. The **main scene** tells Godot where to start the game.

5 Rename **Node2D** to **Main.** Notice in the top middle of the editor that the new **scene** is **unsaved**. Press **CTRL + S** to save it as the **main scene** of the project.

Do **not** click **create** yet.

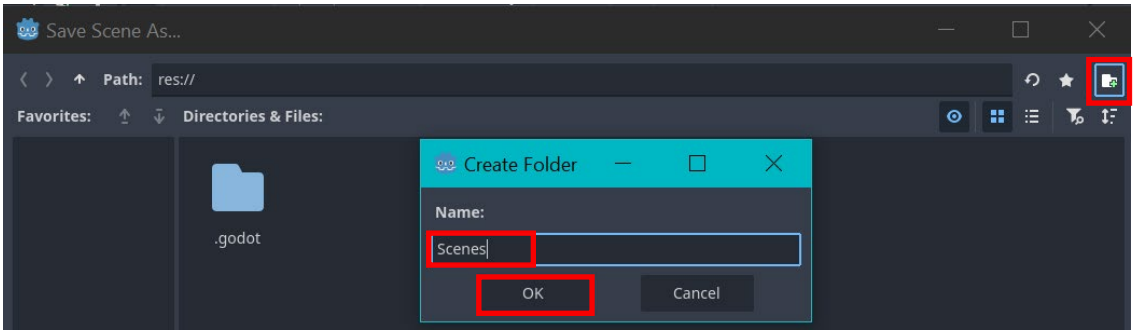


Reminder:

Rename a node by **double-clicking** it.

6 Lots of files will be used throughout this project. Let's create folders to stay organized.

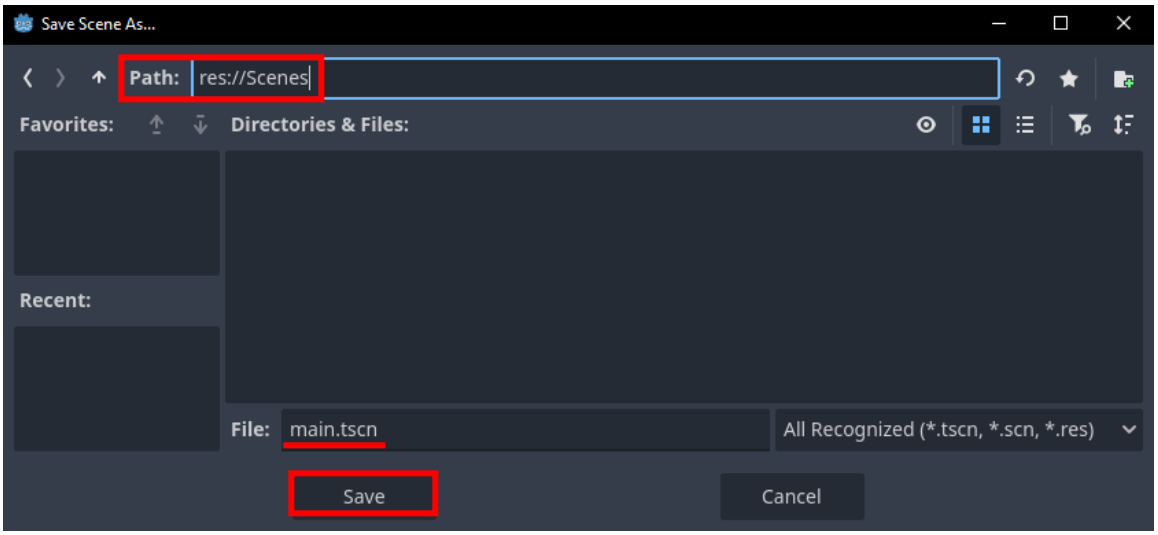
In the top right corner of the **Save Scene As** window, select the **folder** icon and name the folder **Scenes,** then click **OK.**



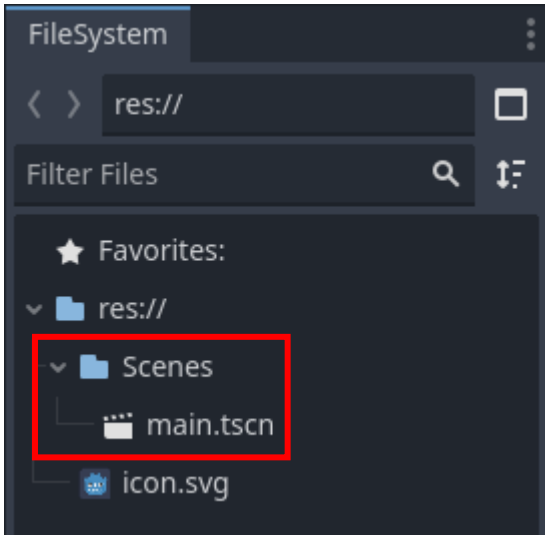
7 The **main scene** will be stored inside the **Scenes** folder. At the top of the window, the path should be the same as shown in the image.

Notice at the bottom of the window, the file should already be named **main.** Godot automatically names scenes the same as the Main root node.

Once the path and file name are checked, click **Save.**



8 At the bottom left of the editor, find **FileSystem.** Double-check the folder structure. The **main** scene should be saved inside the **Scenes** folder; make sure the folder hierarchy is the same as shown in the image.

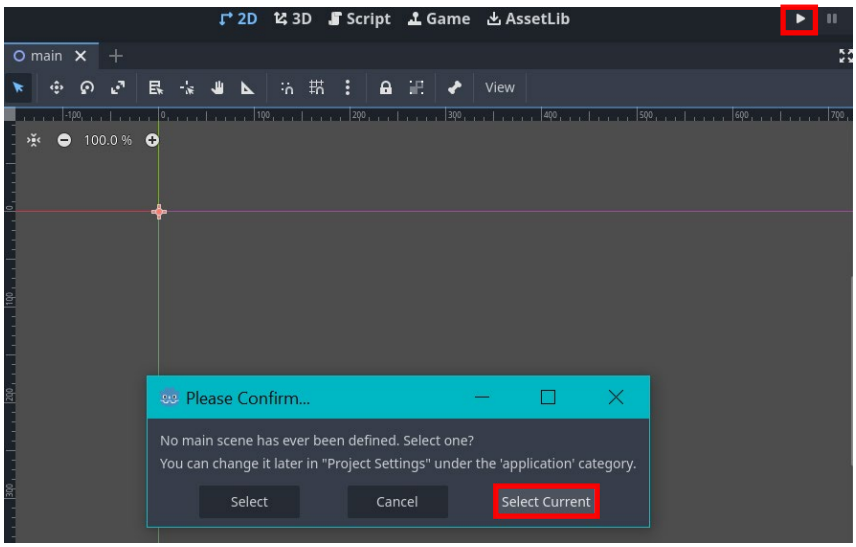


Pro Tip:

Click on the arrow next to each folder to view their contents.

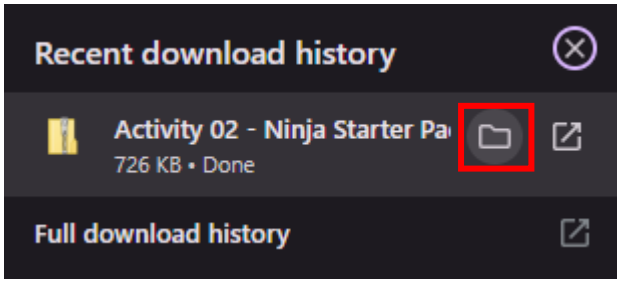
9 One last thing to set up the main scene. In the top right corner, click the **play** button to run the game.

Click **Select Current** to define the main scene, then close out of the playtest window once it appears.



Notice that it's just a gray screen right now; that's because nothing has been added to the scene yet!

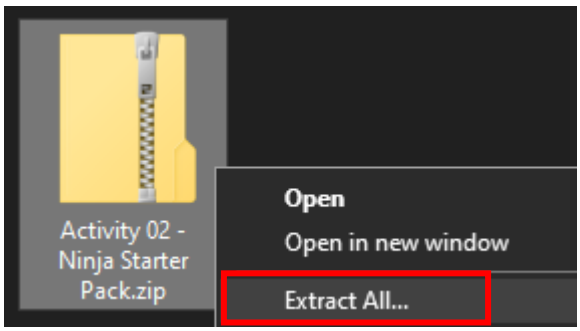
10 Download **BB Activity 02 – Ninja Starter Pack.zip**; a **Downloads** window should pop up at the top right of the screen. Click on the **folder icon** to open the location of the file.



Reminder:

File Explorers can be reopened in the browser by pressing CTRL + J on the browser tab and clicking the folder icon.

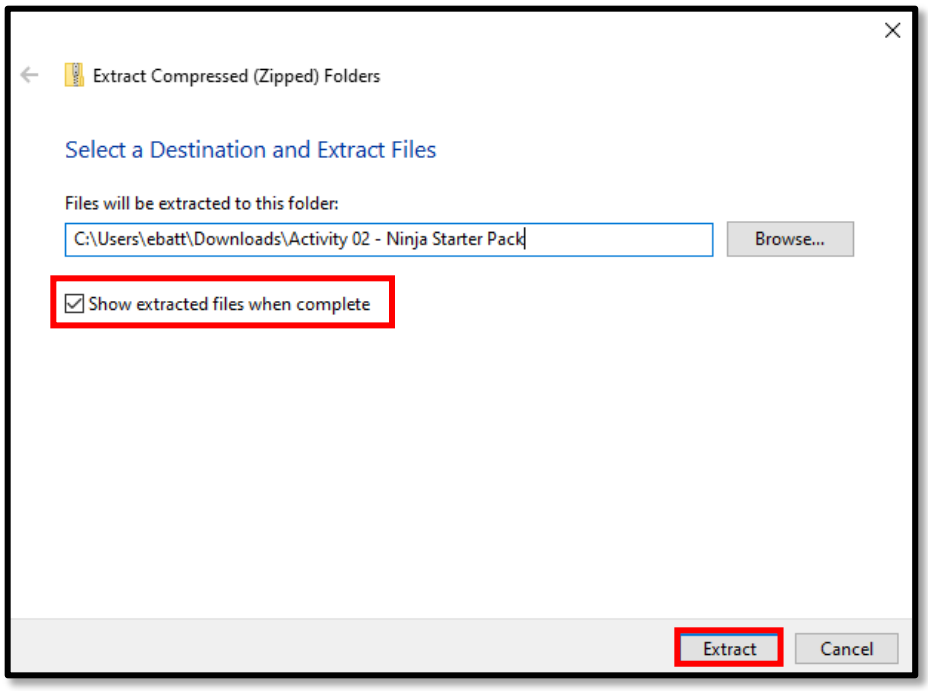
11 Right-click on the **zip** file and select **Extract** or **Extract All**.



Reminder:

Files can't be opened when they are zipped in a folder. To unzip a folder, it must first be extracted.

12 Make sure the **Show extracted files when complete** is checked, then click **Extract.** This will create another **File Explorer** window of the extracted files; do **not** close that window.



Pro Tip:

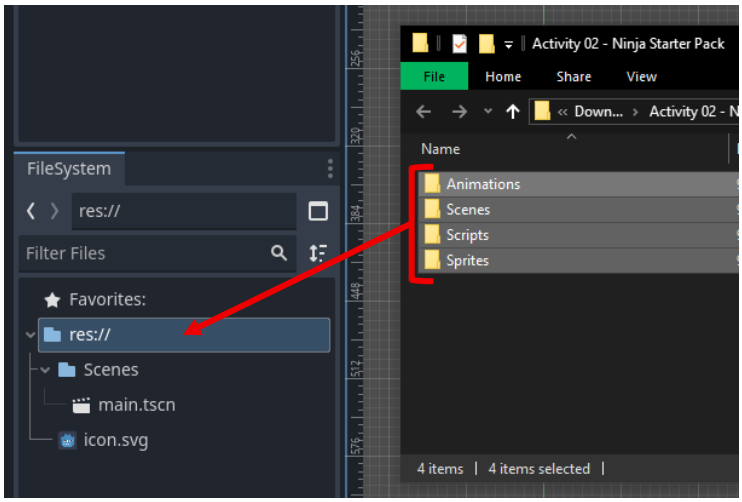
The path will look slightly different from the picture shown because all computers have their own username!

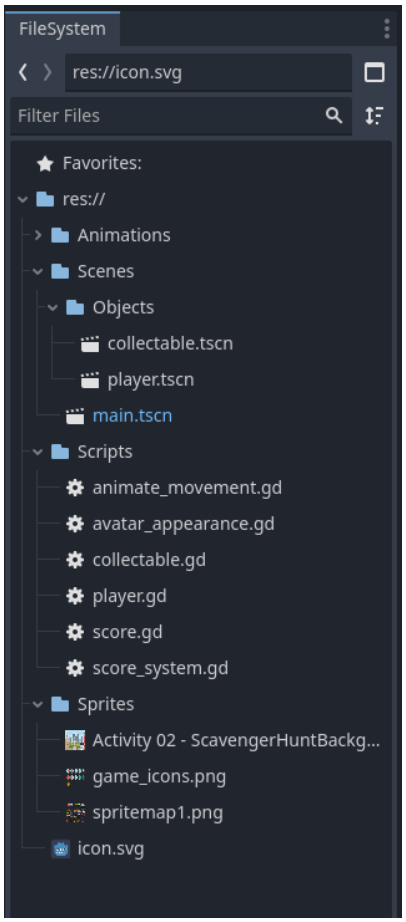
13

Make sure that Godot and the **File Explorer** (with the extracted files) are both open. Check that in Godot's **FileSystem**, the **res://** folder is selected.

In the **File Explorer,** press **CTRL + A** to select all the folders and drag them into **res://** as seen in the image. Check that the following files and folders are present.

Ignore the console errors; these bugs will be fixed at a later point.





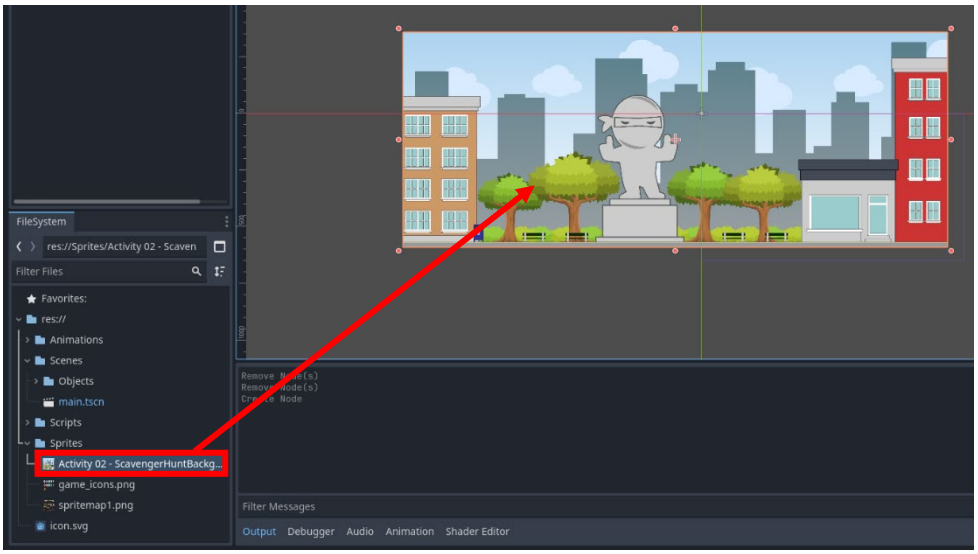
Pause for **Sensei Stop #2**!

Check in with a Code Sensei before moving on. Make sure the main scene is set up and all files are imported correctly.

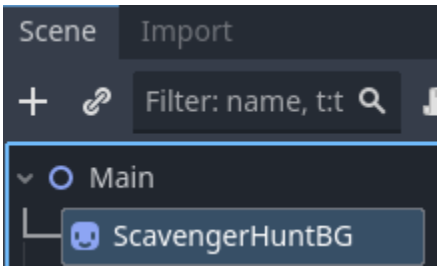
Reminder: Press **CTRL + S** to save your work!

14 It's time to add the background image to the game!

In **FileSystem,** navigate to **res:// > Sprites** to drag and drop **Activity 02 – ScavengerHuntBackground** into the game window.



This will automatically create a **Sprite2D** node named **Activity02-ScavengerHuntBackground.** Rename the node to **ScavengerHuntBG.**

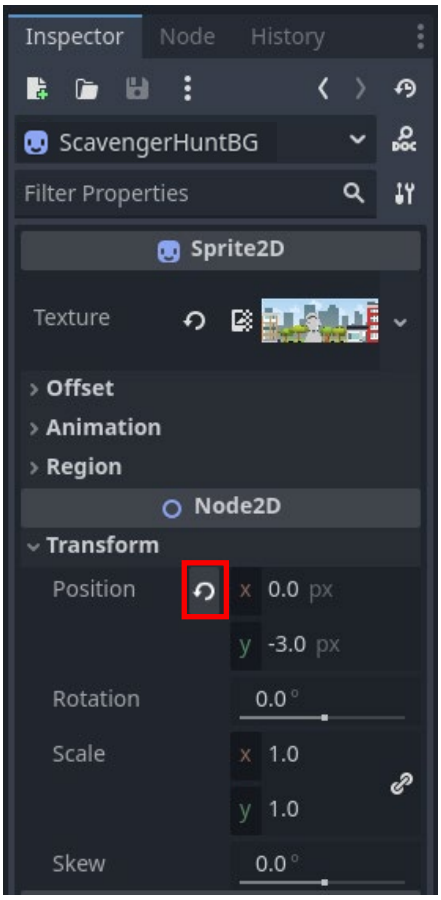


Pro Tip:

Zoom out in the editor by scrolling down on the mouse to see the entire background image.

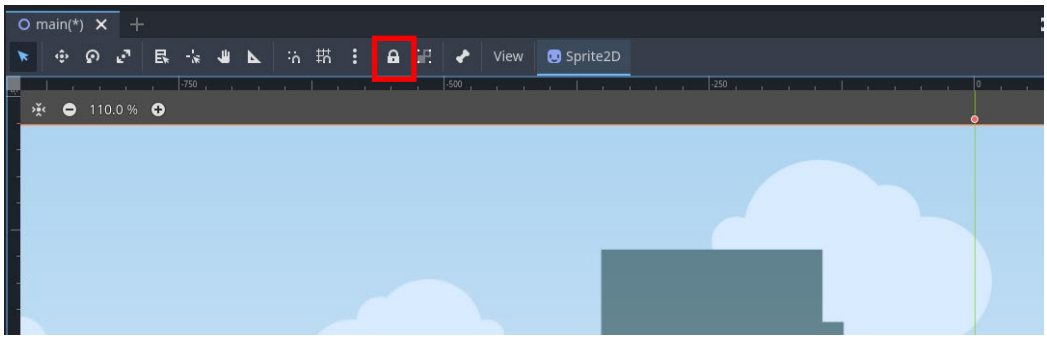
15 Check that the background image is centered.

In the **Inspector** of **ScavengerHuntBG**, open the **Transform** drop-down menu**.** Next to the **x** value, select the **reset** symbol. This resets the **x** and **y** values to **0**.



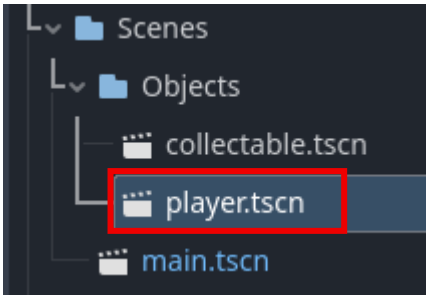
16 It's important that the background stays in this position.

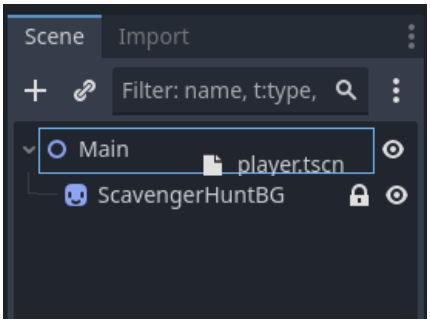
At the top of the game window, select the **lock** icon. This will lock the background in place and prevent it from accidentally moving when the mouse is dragged over it.



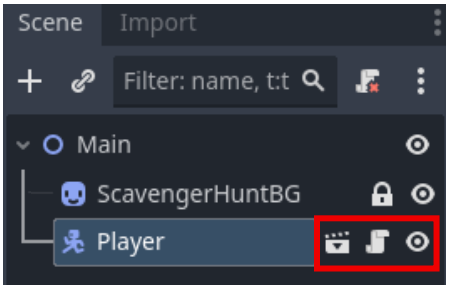
This will help avoid selecting the Background when the Player, Platforms, and Collectables will be moved and resized around the scene.

17 Open the **Scenes** folder, and open **Objects.** Find **player.tscn** then drag and drop the **player scene** onto the **Main root.**





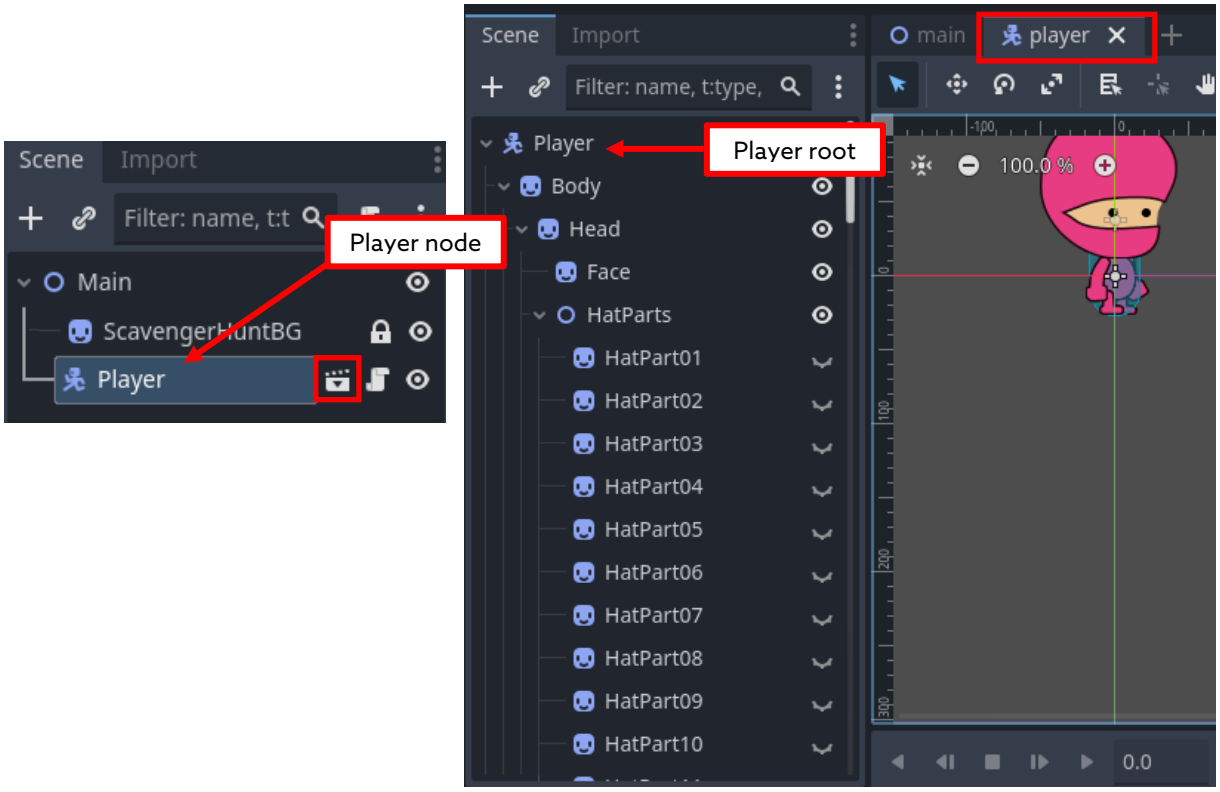
18 Look around the **Scene** menu, what icons does the **Player** already have?



19 The Player has a **scene** and a **script** already attached to it.

Click on the **scene** icon. This will open the Player's **scene.**

Without changing anything, explore the scene by scrolling down. Notice the Player has many different nodes and even a collision shape! The Player is a **sub-tree** in the project, where the **Player** node is the *root* of the **player scene** but is a *node* in the **main scene**.

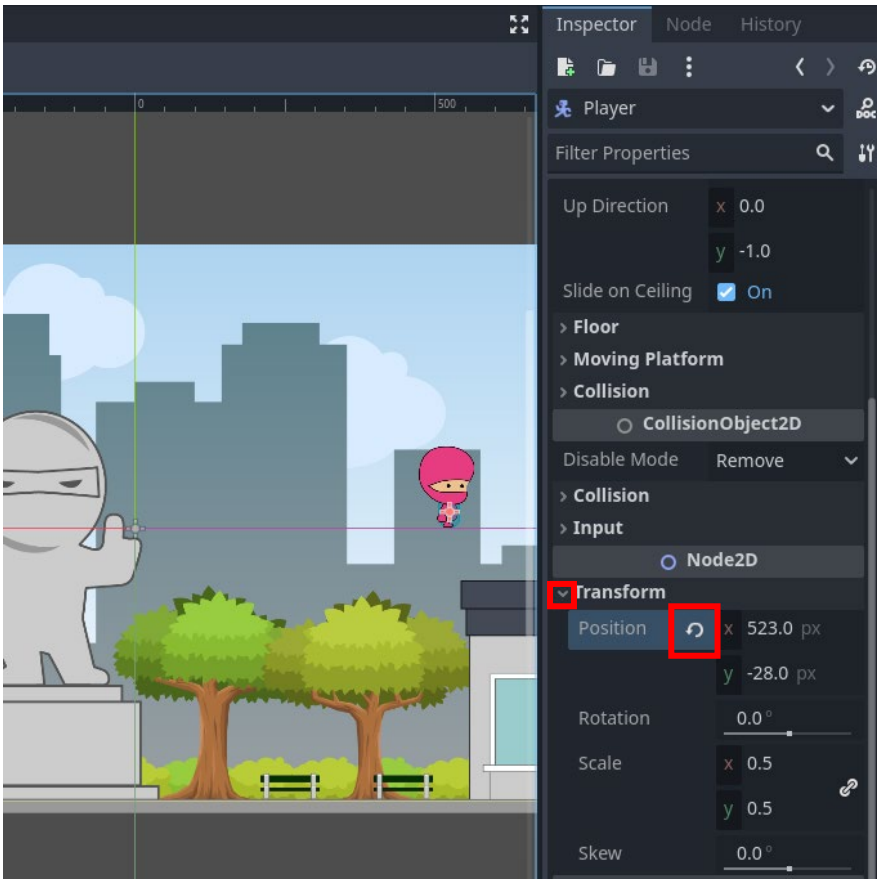


New Concept: Sub-tree

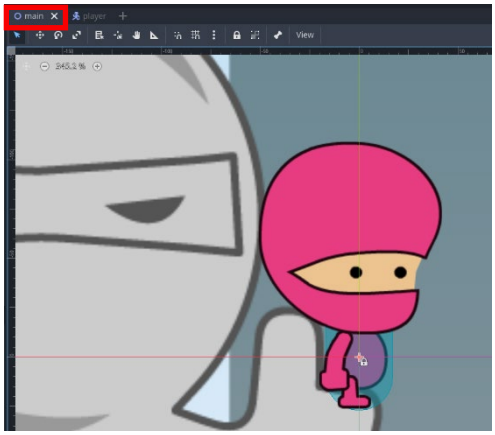
A **sub-tree** is a section of a tree with its own root and children. The Main root is the first node created in the project, but each node in a tree can be considered as the root to its own sub-tree.

20 Navigate back to the **main scene.** The Player should appear *in the center* of **ScavengerHuntBG**.

If not, reset its position in the **Inspector**. Open **Transform** and click the **reset** button.



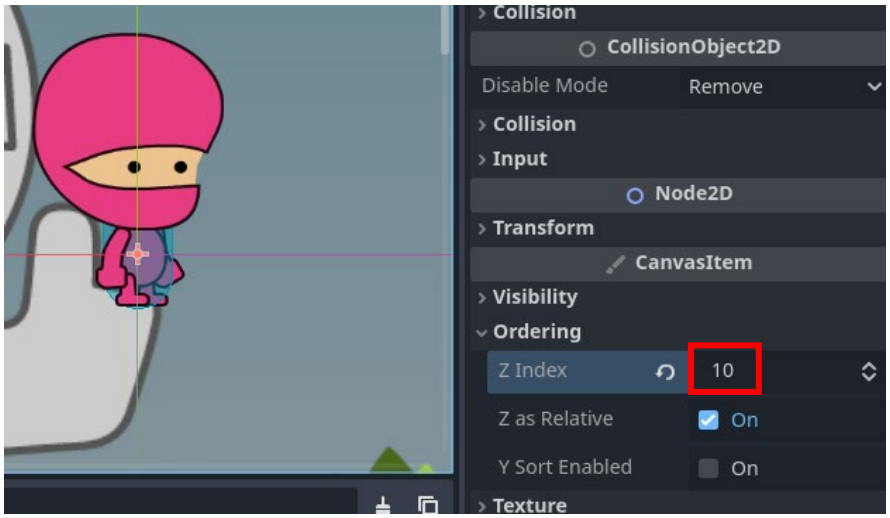
In the game window, zoom in on the **Player.** Is the player missing anything?



21 Some of the player's body parts are missing! Think for a moment about why this might be.

They seem to disappear into the background. This is because Godot in 2D has Z layering, too (not just in 3D!). This means that the player's other arm and leg are *behind* the background sprite. Time to fix this!

Select the **Player** node. In the **Inspector**, go to the **CanvasItem** section and open the **Ordering** menu. Set the **Z Index** to **10. ScavengerHuntBG** has a Z Index of 0, so all parts of the Player will be placed on top of it by nine Z values.



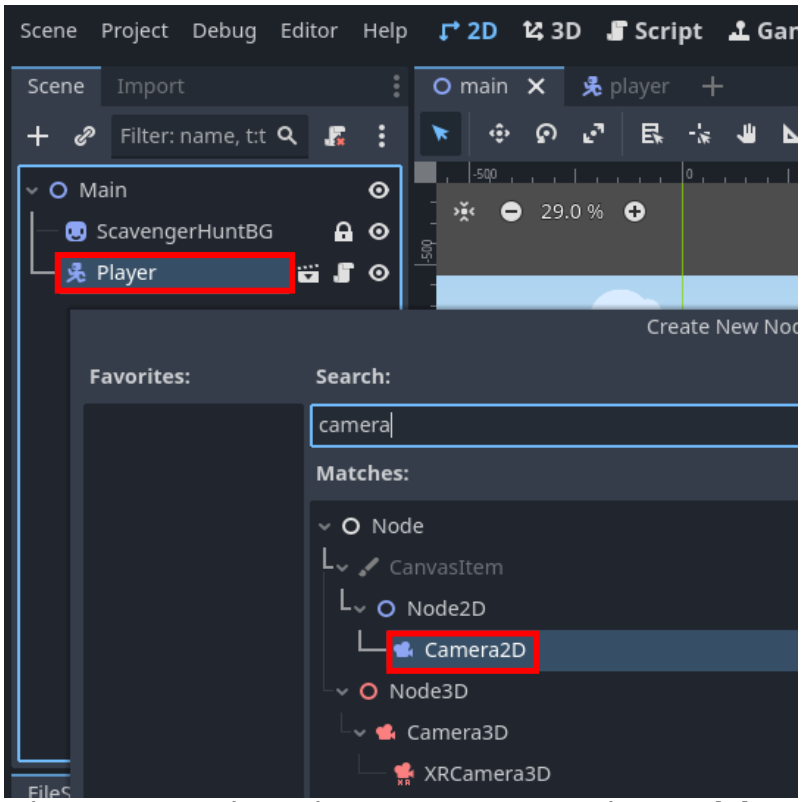
22 In the top right corner, click the **play** button to run the game. What happens?

Hmm, something seems wrong. The Player goes off screen and disappears, even though it should be **always visible**.

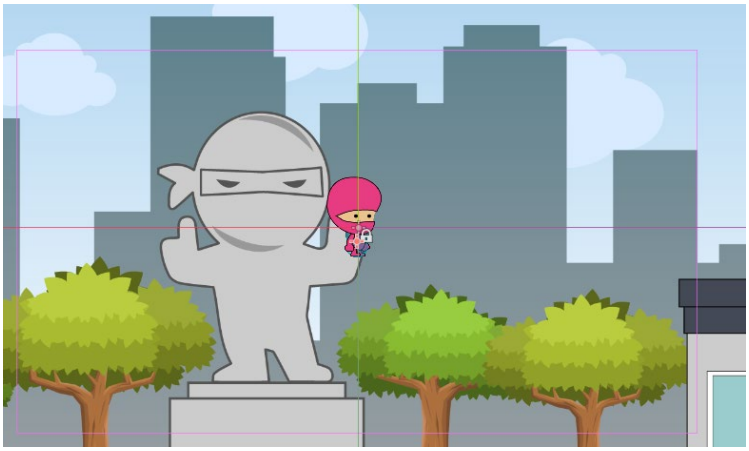
23 Add a **Camera2D** node as a child node to the Player.

Select the **Player** and press **CTRL + A** to create a new node.

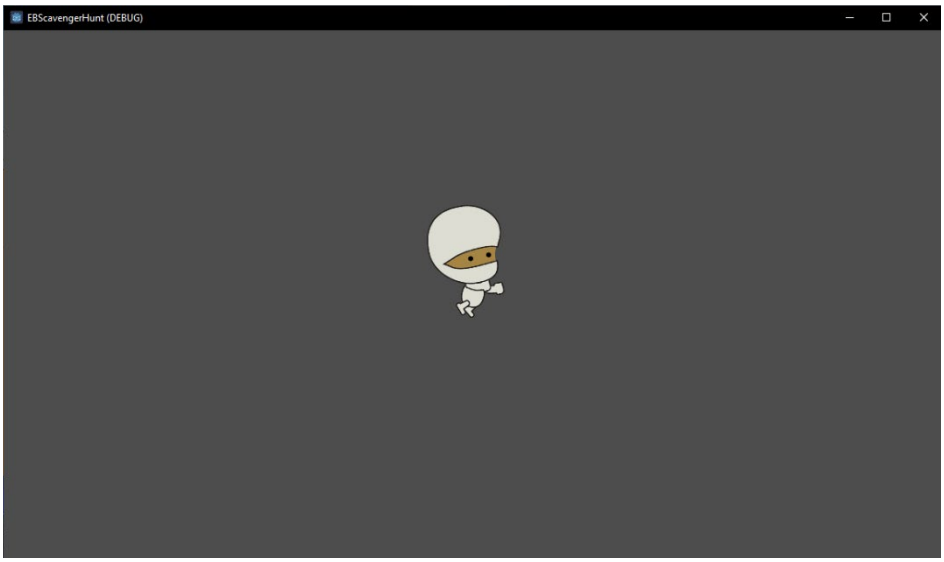
Search for **"camera",** select **Camera2D,** and press **Create**. Now, a **Camera2D** node, inside of the **Player** node, is in the **Main Scene**.



Notice that in the game window, the main scene now has **pink lines** around the **Player,** outlining the camera's boundaries.



24 **Playtest** the project again by clicking the **play** button in the top right.

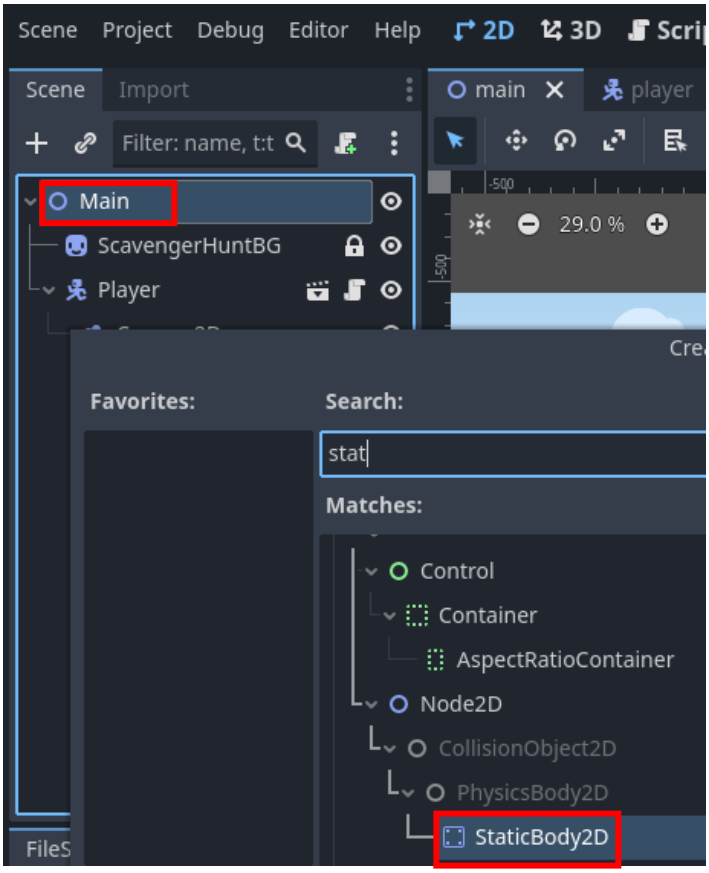


Great! The **Player** is always visible. However, the **Player** still quickly falls through the ground.

25 Prevent the player from falling through the ground. Add a **StaticBody2D** node as a child node to the **Main root.**

Select **Main** and press **CTRL + A** to create a new node.

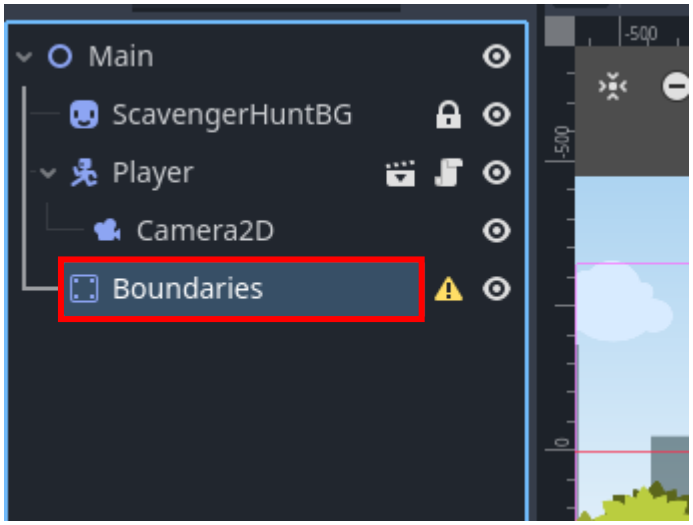
Search for **"static"**, select **StaticBody2D,** and press **Create**. Check that the hierarchy is the same as shown in the image.



Reminder:

Nodes can be dragged and dropped to be rearranged.

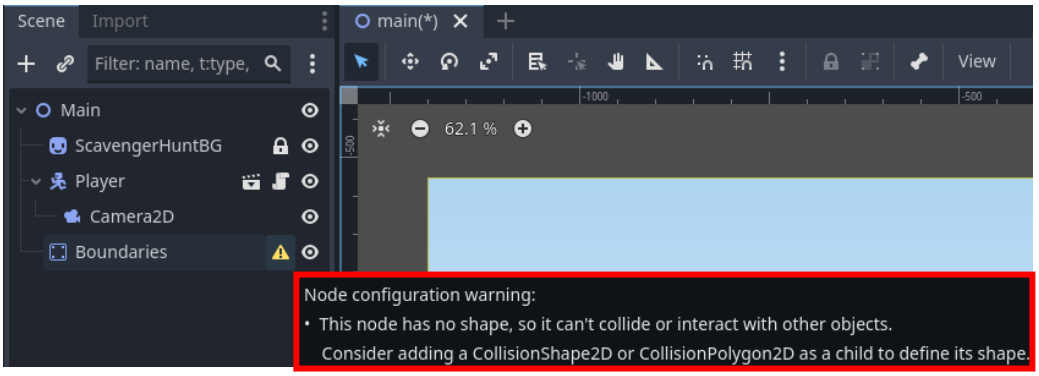
26 Rename the **StaticBody2D** to “Boundaries”.



Notice the yellow warning sign next to the StaticBody2D node. How might this be removed?

27 Think back to what the hazard symbol means from Dropping Bombs, then hover over the symbol.

The **Boundaries** node has a warning sign because it still needs a Collision Shape!

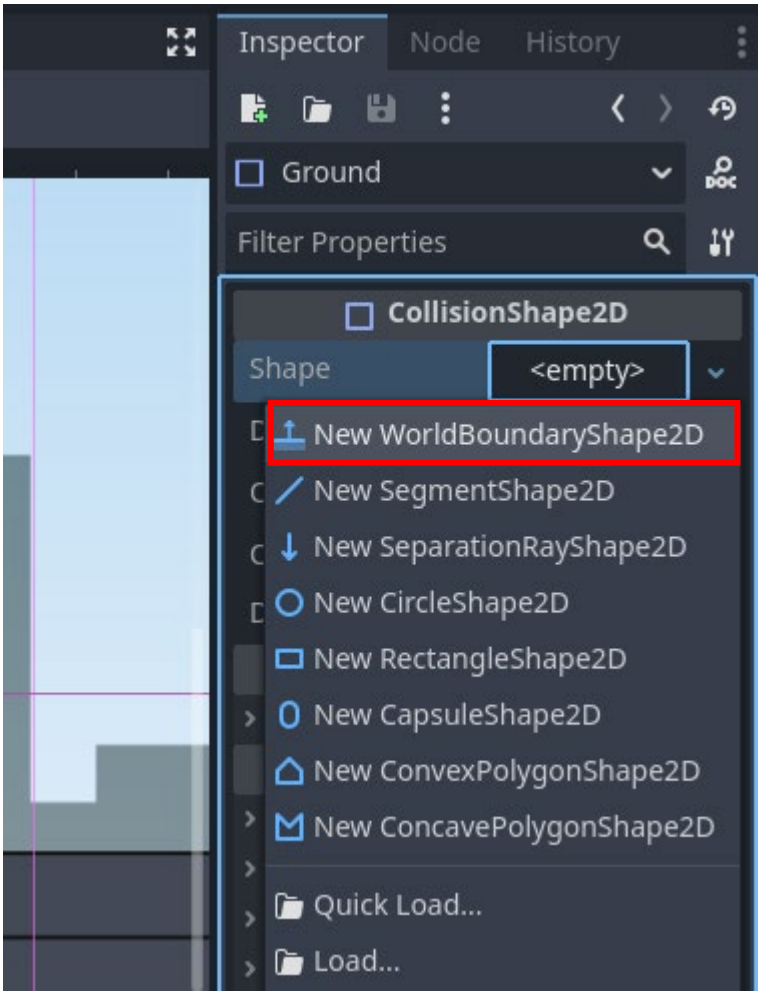


Add a **CollisionShape2D** node as a child node to **Boundaries.** Rename the **CollisionShape2D** to "**Ground**".

Reminder:

Press **CTRL + A** to create a new node.

28 In the **Inspector** menu, set Ground's collision shape to **WorldBoundaryShape2D**.



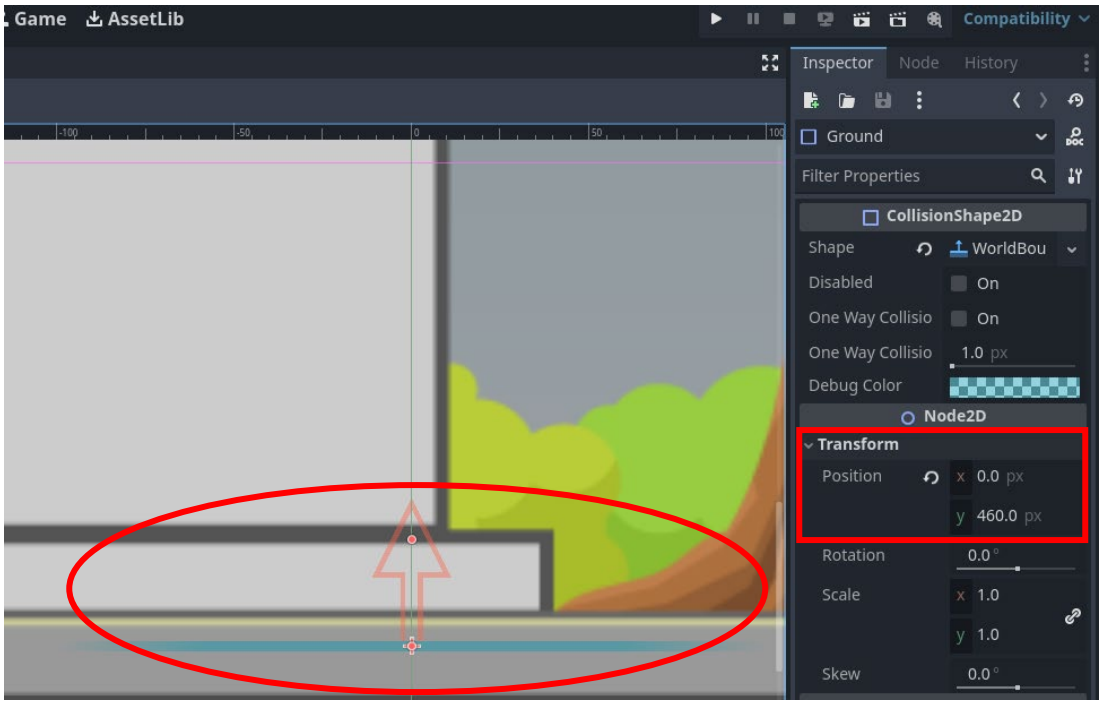
New Concept: WorldBoundaryShape2D

WorldBoundaryShape2D works like an infinite straight line that forces all physics bodies to stay above it. Remember, the Player is a 2D Physics Body, meaning it will be forced to stay above this boundary!

29

In the **Inspector**, set the position of **Ground** to the **y** value shown in the image. This will set the world boundaries at the *bottom* of **ScavengerHuntBG**, in line with the pavement.

Zoom in on the **CollisionShape2D**. Notice an orange arrow pointing up with a blue line that fades away. The **blue line** is the **infinite straight line** that forces all physics bodies to stay "above" it, and the **orange arrow** indicates what is **"***above"* the blue line.



Playtest the project to check that the player does not fall through the ground.

Pause for **Sensei Stop #3**!

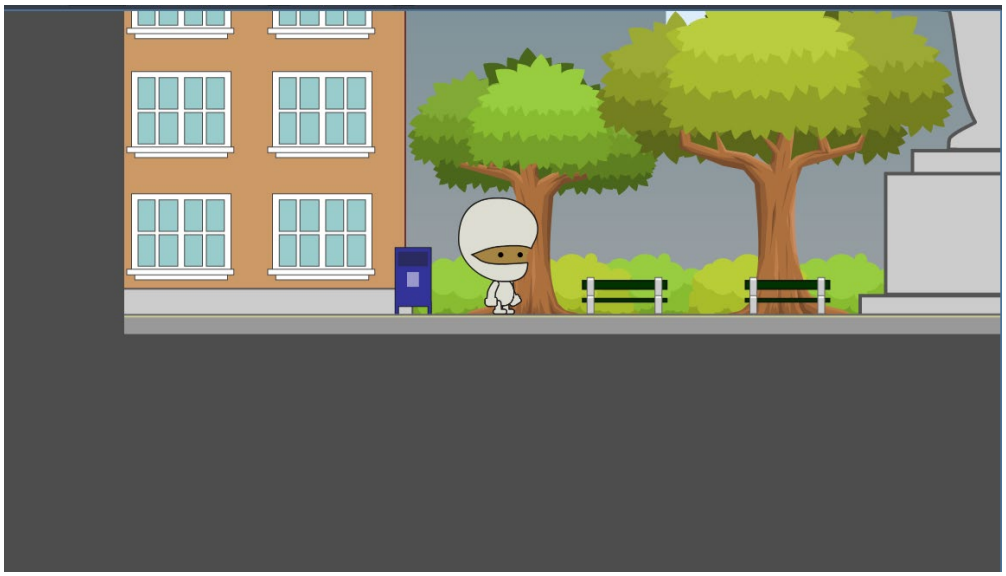
Check in with a Code Sensei before moving on. **Playtest** the project to make sure that the **Background** and **Player** are set up and the **Player** no longer falls through the ground.

Reminder: Press **CTRL + S** to save your work!

30 When the game starts, the player falls. Drag the **Player** so that it is placed on the ground when the game starts.

Reminder: Use the move icon at the top of the editor to reposition the player.

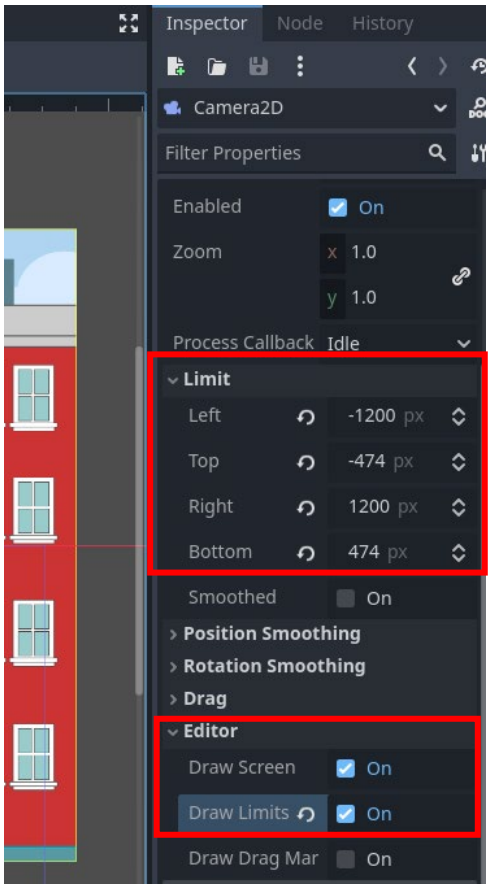
31 Playtest the project again. Notice that the camera can see beyond the background image!



To fix this, the **Camera2D** needs to have limits, so it does not wander too far.

32 Select the **Camera2D** node. In the **Inspector**, scroll down and open the **Editor** drop-down menu**.** Check the **Draw Limits** option **on**. This will create yellow lines which show the camera's limits.

Scroll back up and open the **Limit** drop-down menu. Enter the same limit values as shown in the image.

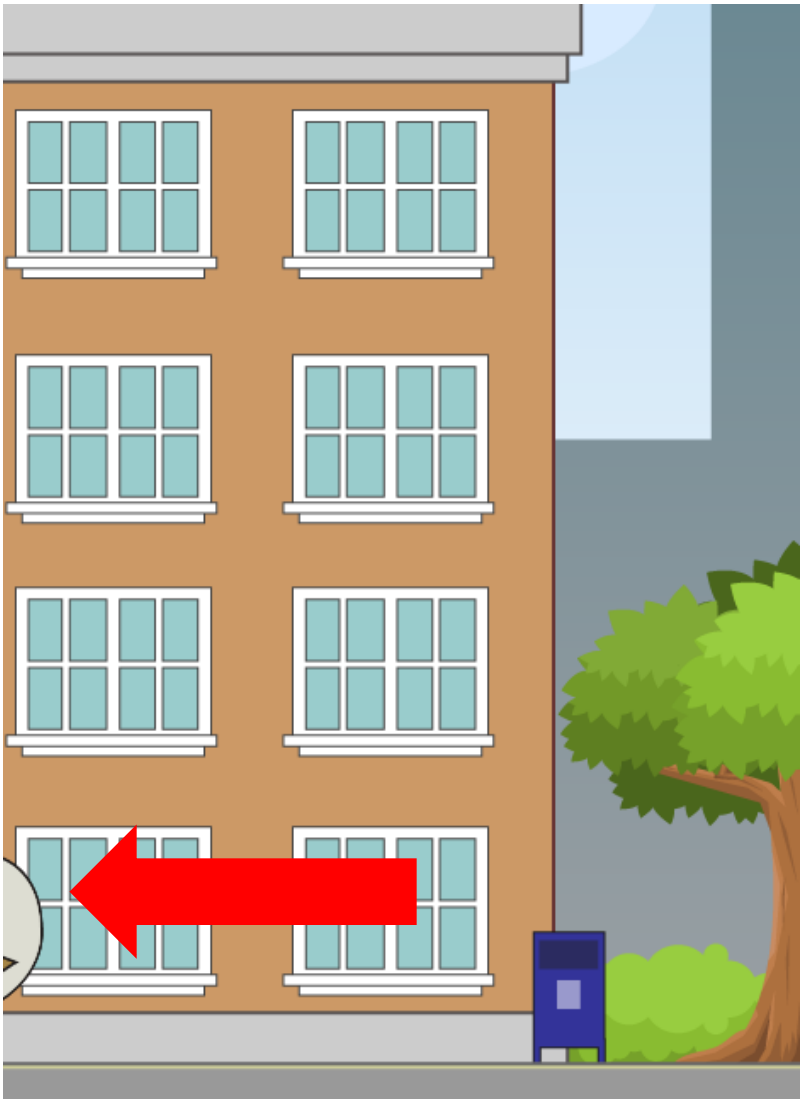


Notice how the yellow lines adjust in the game window as the values change.

The yellow lines should be outlining the **ScavengerHuntBG**'s image!

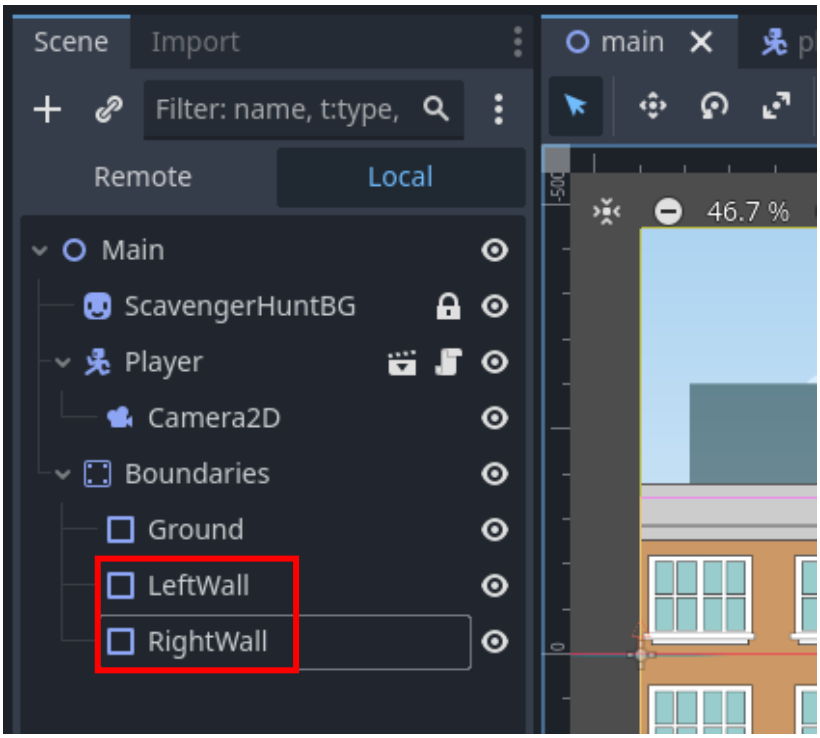
33 Playtest the project. Check if the camera lets you see beyond the background image or not. If you're still able, look back at the previous step.

Try walking to **either edge** of the game using the **arrow keys**. What happens?



34 Uh oh! The player **walks right off** the screen and can't be seen anymore! Add walls to ensure the player stays within the game bounds.

Add 2 **CollisionShape2D**s as child nodes to the **Boundaries** node. Rename the nodes **LeftWall** and **RightWall**.

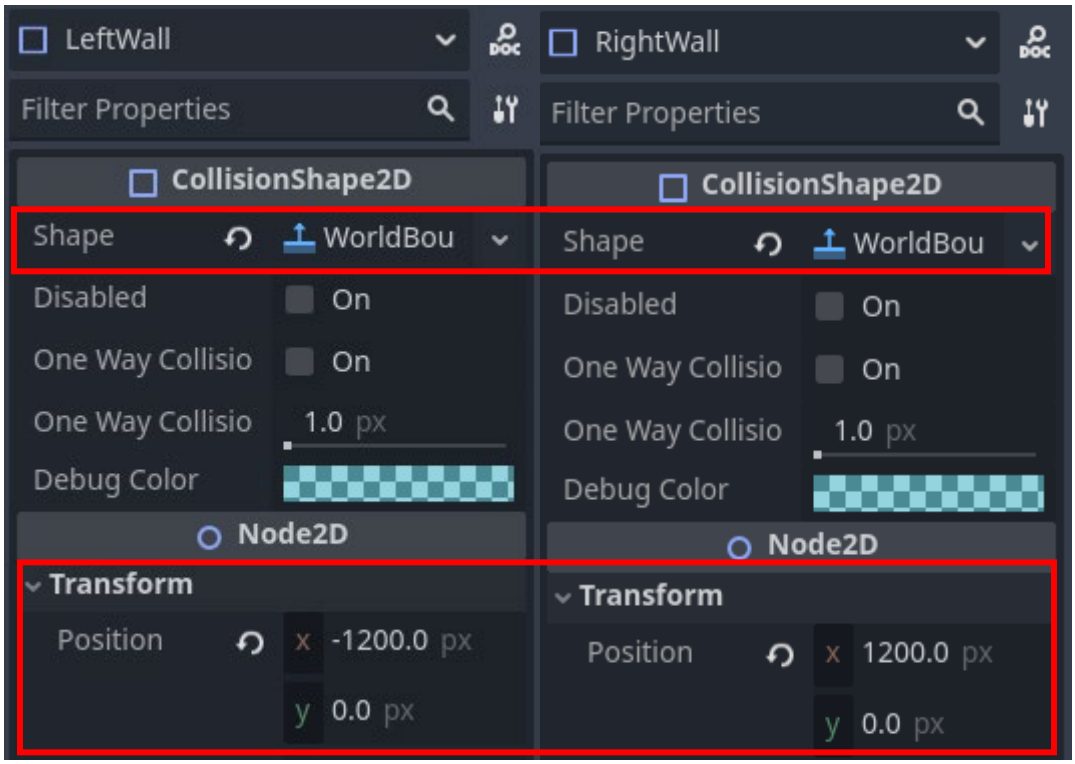


35 Notice more hazard symbols; what needs to be added to remove these? Refer to **Steps 26 – 27** for additional guidance on how to fix this.

Reminder:

Hover over the hazard symbol to see what it means!

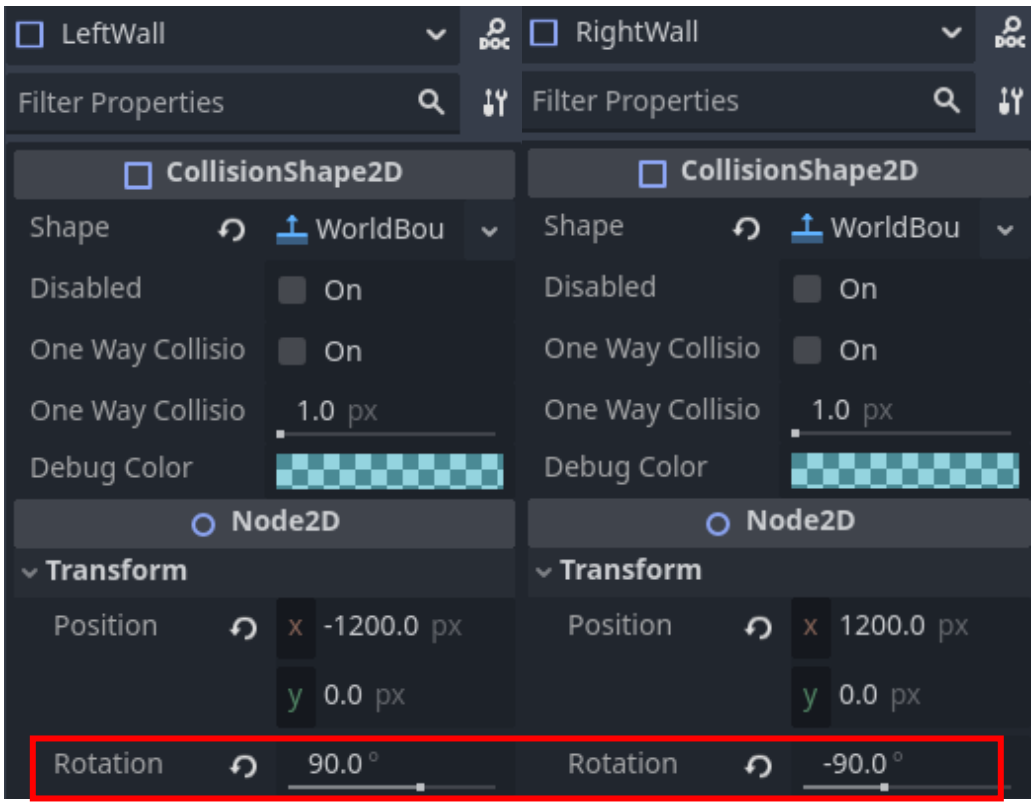
Their shapes need to be set to **WorldBoundaryShape2D**. Set the x-position of **LeftWall** to **-1200** and **RightWall** to **1200**. Double check the values being changed for the *left* and *right* wall to ensure the **x** values are exactly **-1200** and **1200.**



Playtest the game. What happens?

36 The Player is floating in the air! This is because the Left and Right walls are still facing up, so they act as if they were the ground.

Rotate LeftWall by **90** degrees and RightWall by **-90** degrees so that they act as walls, instead of the ground.



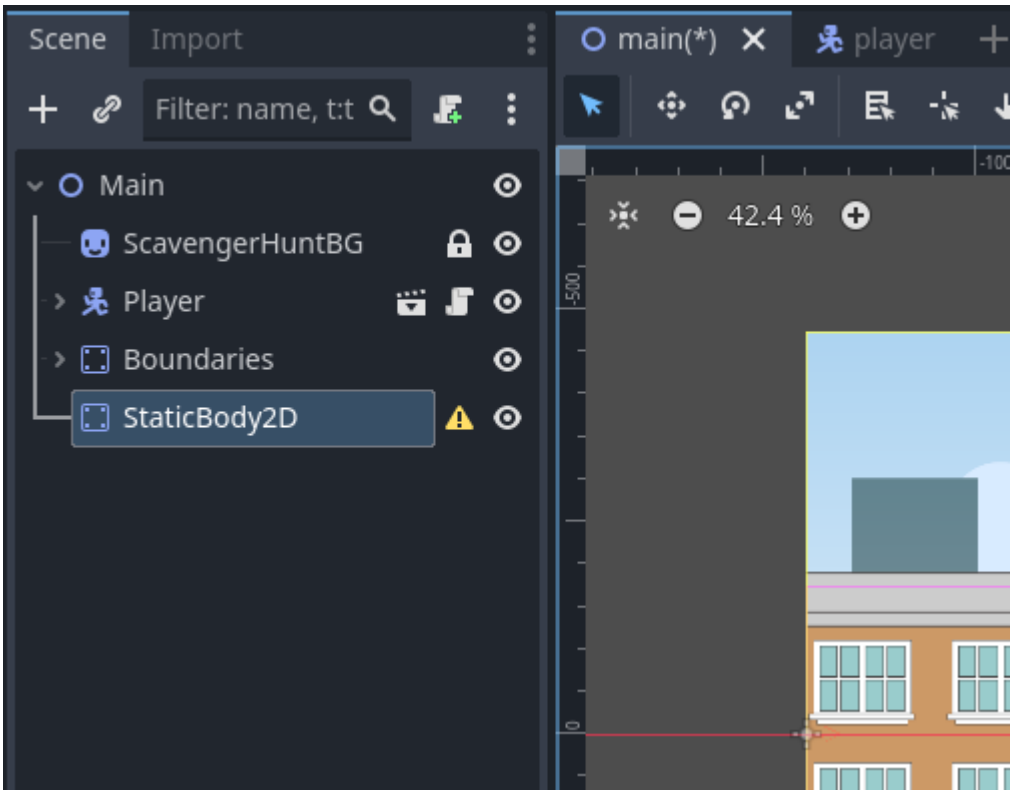
Pause for **Sensei Stop #4**!

Check in with a Code Sensei before moving on. Make sure that the **Camera** is confined properly and the **Player** no longer walks off the sides of the scene.

Reminder: Press **CTRL + S** to save your work!

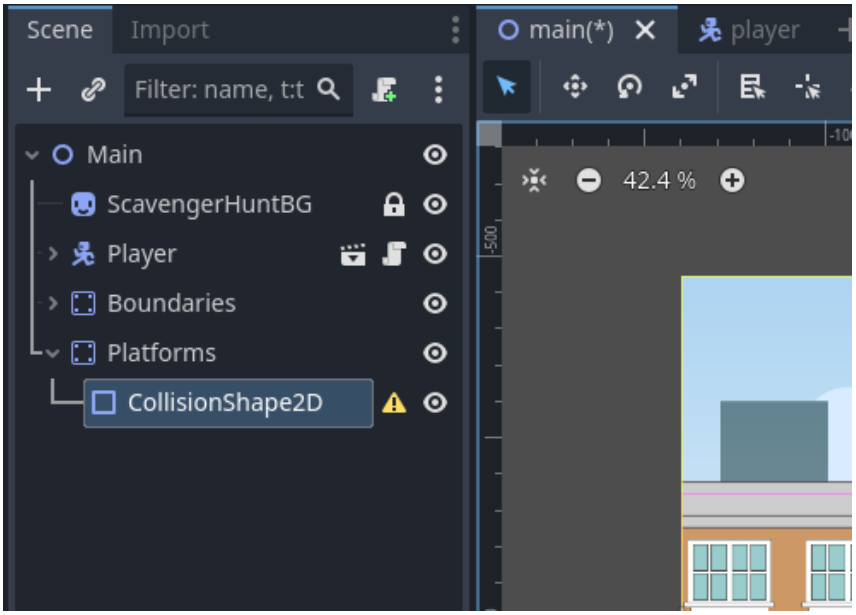
37 To make this project a true Scavenger Hunt, the **Player** needs to be able to explore above just the ground! To do this, **platforms** can be placed around the game for the Player to jump on.

Add a **StaticBody2D** node as a child to the Main root.

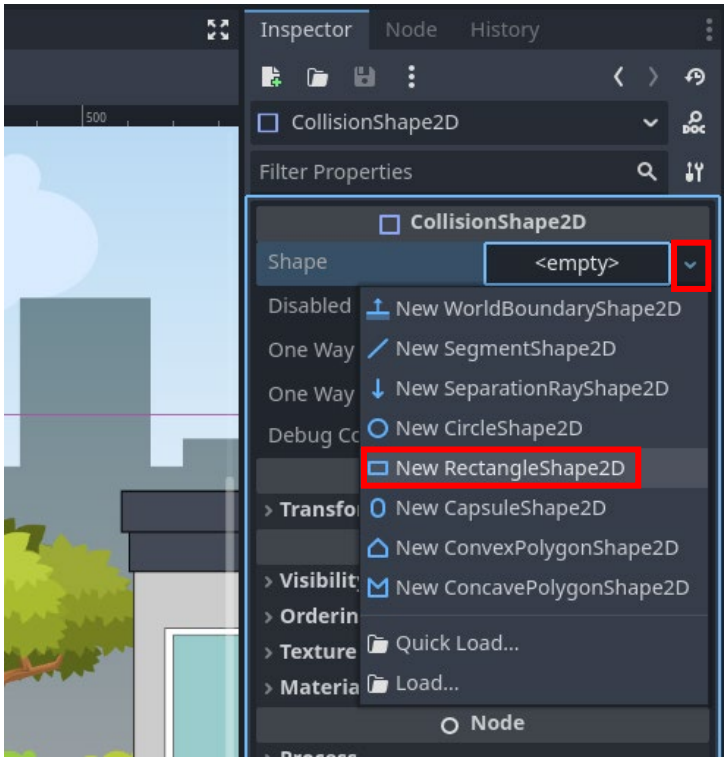


Reminder: The arrows next to nodes can be selected to collapse the node's children.

38 Rename the **StaticBody2D** to **Platforms**, then add a **CollisionShape2D** as a child node to **Platforms**.



39 With the **CollisionShape2D** selected, go to the **Inspector** menu. Select a shape that would best fit a windowsill.

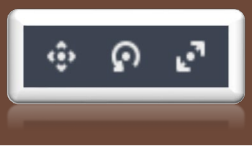


40 **Drag** the **CollisionShape2D** around and **resize** it to match one of the tan building's windowsills.

Don't use **scale mode** to resize, as it resizes the entire **CollisionShape2D**. Instead, drag the red dots to resize the **RectangleShape2D** that the CollisionShape2D uses.

Reminder:

At the top of the game window, these tools can be used to move, rotate, and resize objects. Hover over them to see their description!

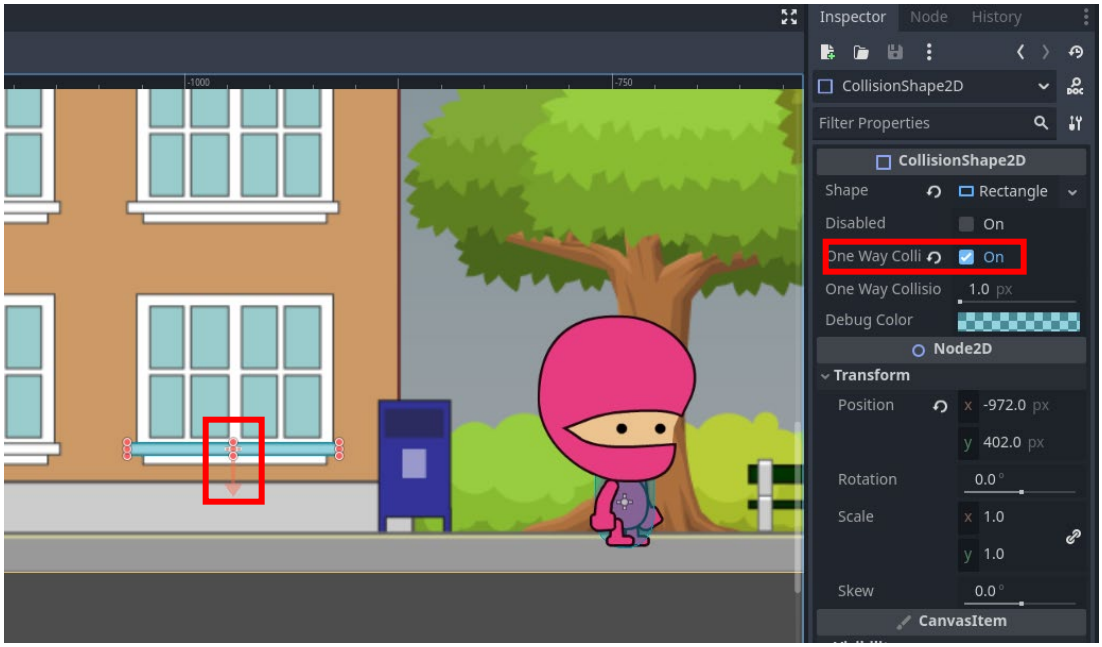


41 **Playtest** the project. What happens when the Player walks towards the platform?

The Player collides with the platform and **cannot walk through it**. This is fine for walls, but platforms should allow a player to **walk through** and **land on top**.

42 Luckily, Godot has a way to address this. Select the **CollisionShape2D** node and go to the **Inspector**.

Select the **One Way Collision** checkbox to enable it. Notice the red arrow that appears, pointing down on the **CollisionShape2D**.



If it's not pointing down, the **CollisionShape2D** may need to be manually rotated.

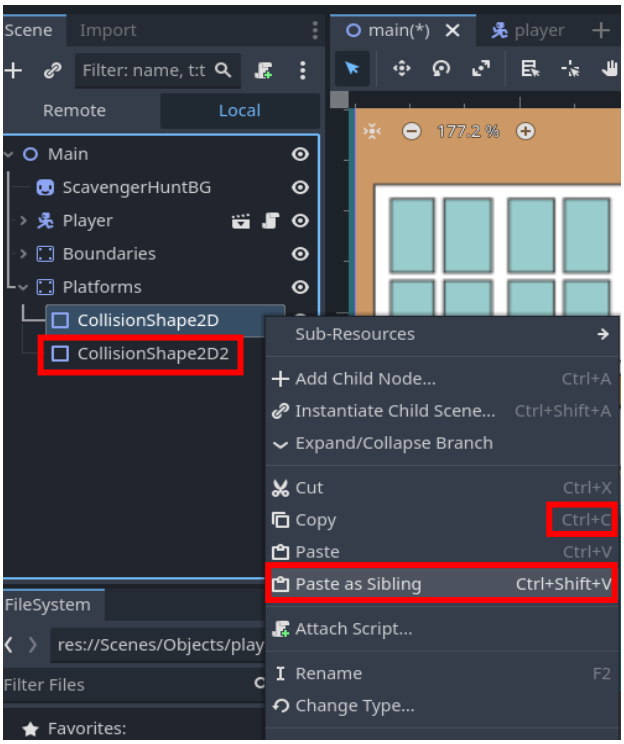
Pause for **Sensei Stop #5**!

Check in with a Code Sensei before moving on. Make sure that the Player can both **walk through** the platform and **land on top** of it.

Reminder: Press **CTRL + S** to save your work!

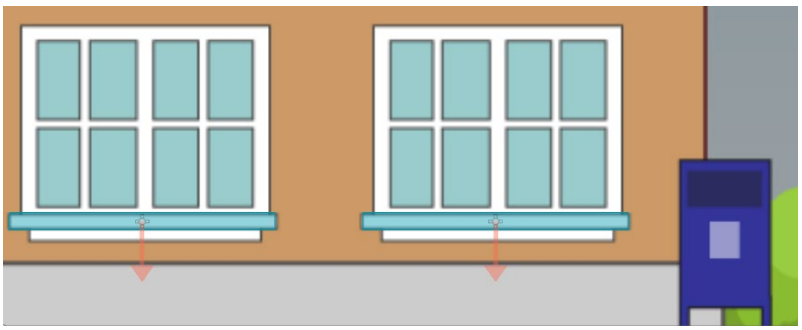
43 One platform isn't enough to explore the whole map! There needs to be many more platforms that fill up the world so the Player can explore.

Select **CollisionShape2D** and press **CTRL + C** to copy it. Right-Click the same CollisionShape2D and select **Paste as Sibling**.

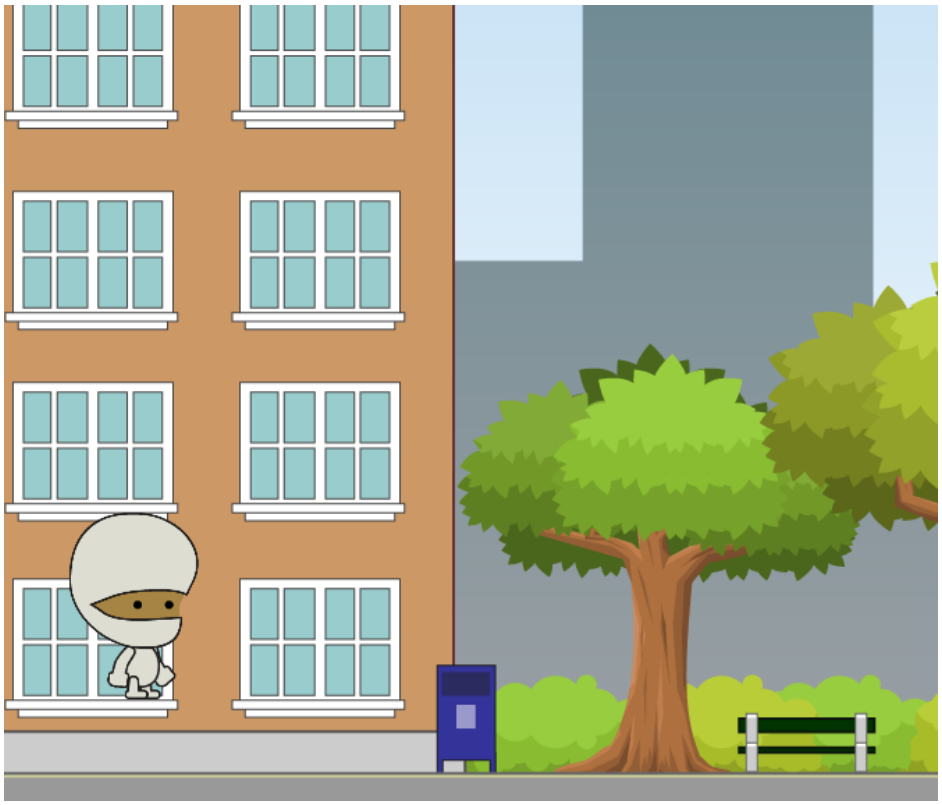


Another **CollisionShape2D** should appear as a child to Platforms called **CollisionShape2D2**.

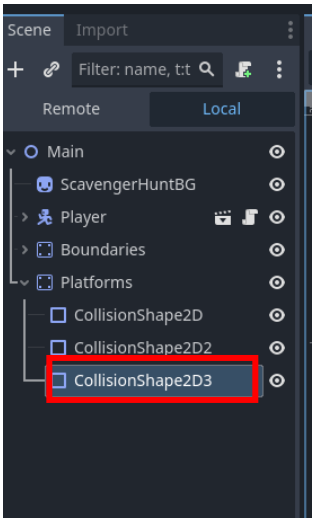
44 The new **CollisionShape2D2** will be placed in the **same position** as the original **CollisionShape2D**. Drag the new one to the **next windowsill** on the left, as shown in the image.



45 **Playtest** the project. Check that the new platform works as expected. If not, refer to the previous steps for help.

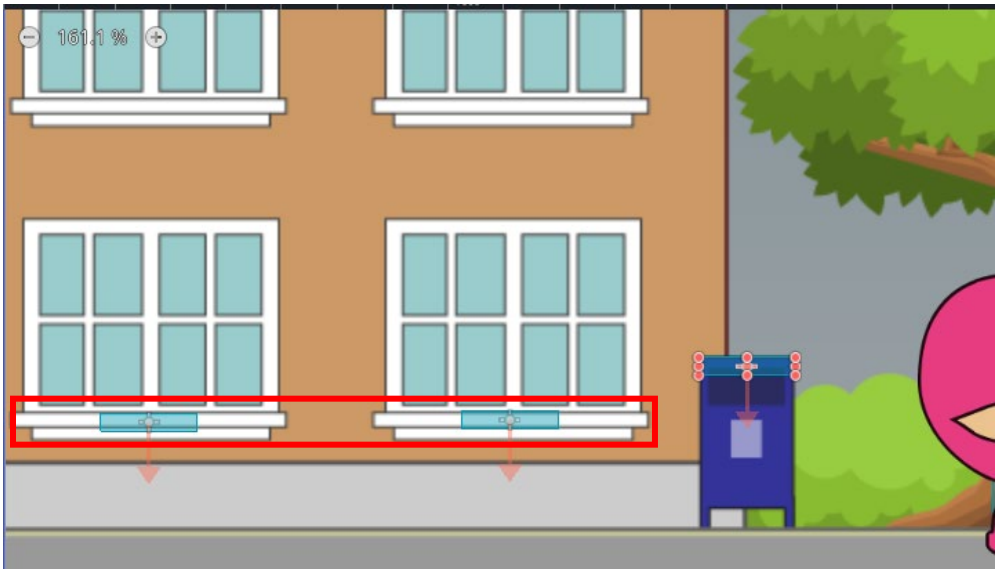


46 Select either of the **CollisionShape2D** platforms and repeat **Step 46** to add a new platform.



47 Reposition and resize the new platform, so that it's placed on top of the mailbox.

Notice how resizing the new platform will change the shape of the other **CollisionShape2Ds**.

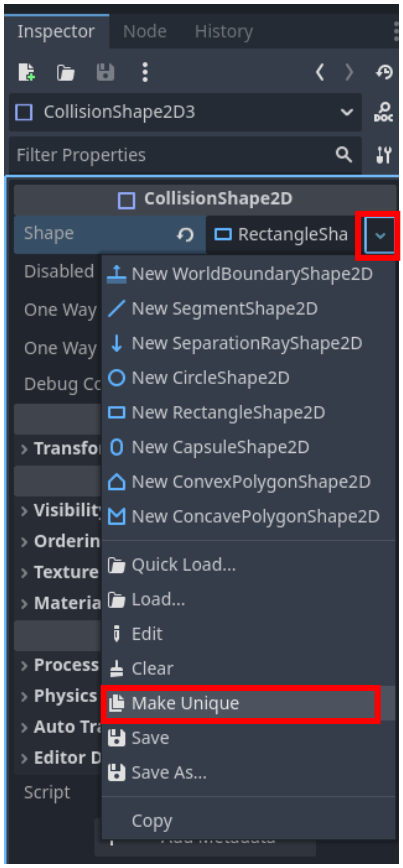


Pro Tip:

This is because all the CollisionShape2D nodes are using the **exact same RectangleShape2D**. Resizing one **RectangleShape2D** will resize the **RectangleShape2D** for all of them.

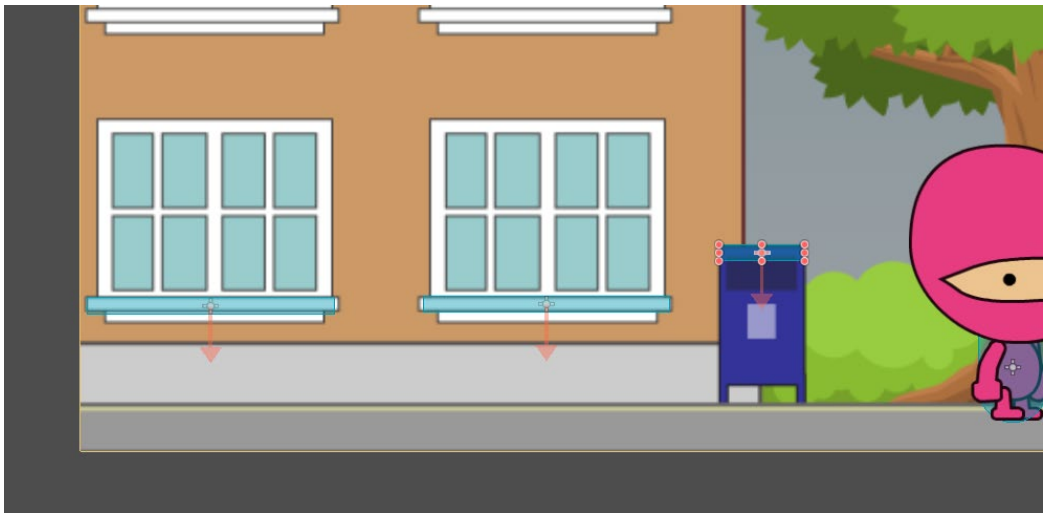
48 Assign a unique platform to some platforms. Go to the **Inspector** and select the **drop-down arrow** next to RectangleShape2D.

Towards the bottom, select **Make Unique**. This gives the selected CollisionShape2D its **own RectangleShape2D** that no other platforms can use.



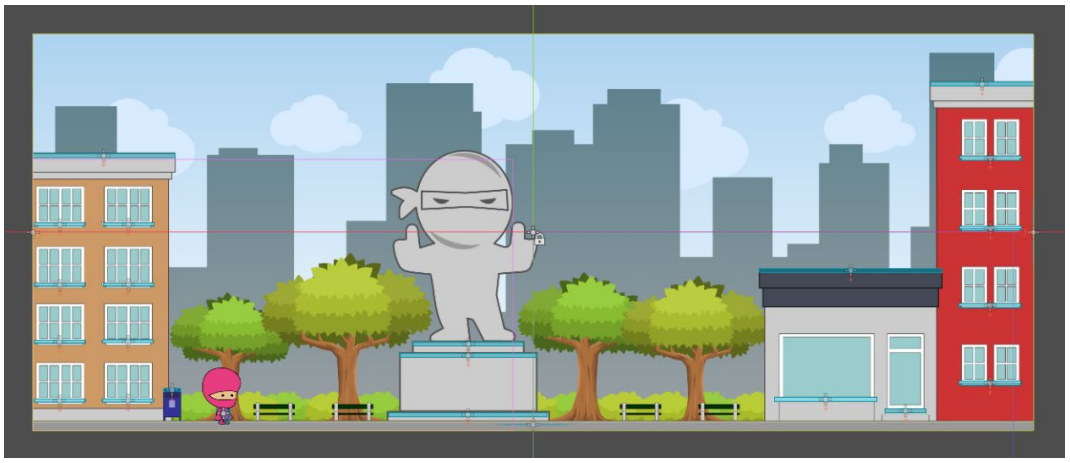
Resize the other platforms back to their original sizes to fit the windowsills. They don't need to be made unique because their **RectangleShape2Ds** should be the exact same size.

49 **Playtest** the project. Check to see if the Player can jump on top of the **mailbox** and the two **windowsills**.



50 Add the rest of the platforms to the scene, as shown in the image.

Notice that not every single platform needs to be made unique. Many platforms can keep the same shape, like the windowsills.



Reminder:

To quickly create duplicates, press **CTRL + D**. Undo mistakes by pressing **CTRL + Z**.

Pause for **Sensei Stop #6**!

Check in with a Code Sensei before moving on. Make sure **all platforms** have been placed onto the scene and are resized correctly.

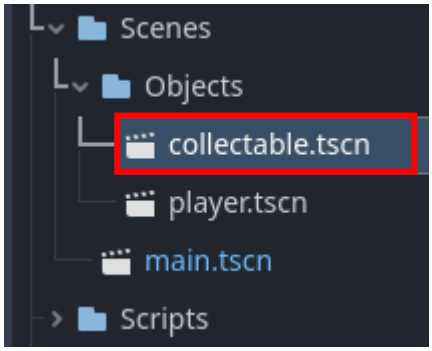
Reminder: Press **CTRL + S** to save your work!

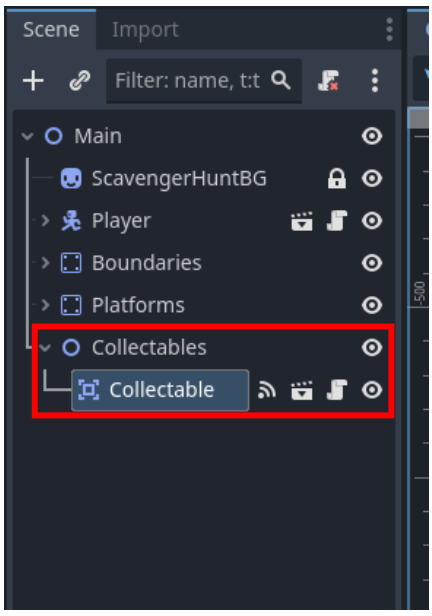
51 Now that there are platforms that the Player can use to explore, there should be collectables for the Player to retrieve.

Add a new **Node2D** as a child node to the **Main root**. Rename it **Collectables**.

In **FileSystem**, locate the collectable scene **collectable.tscn.** Remember, scenes that are not main scene are stored in the **Objects** folder.

Drag **collectable.tscn** onto the new **Collectables** node.

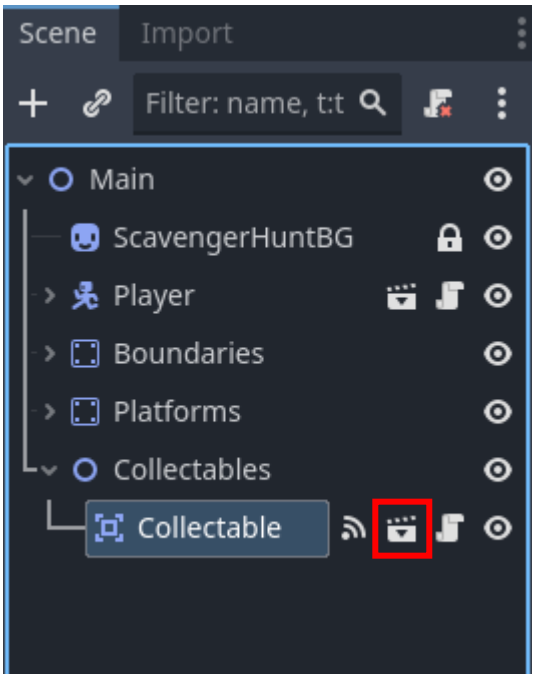




Reminder:

Folders can be opened by clicking on the arrow next to them!

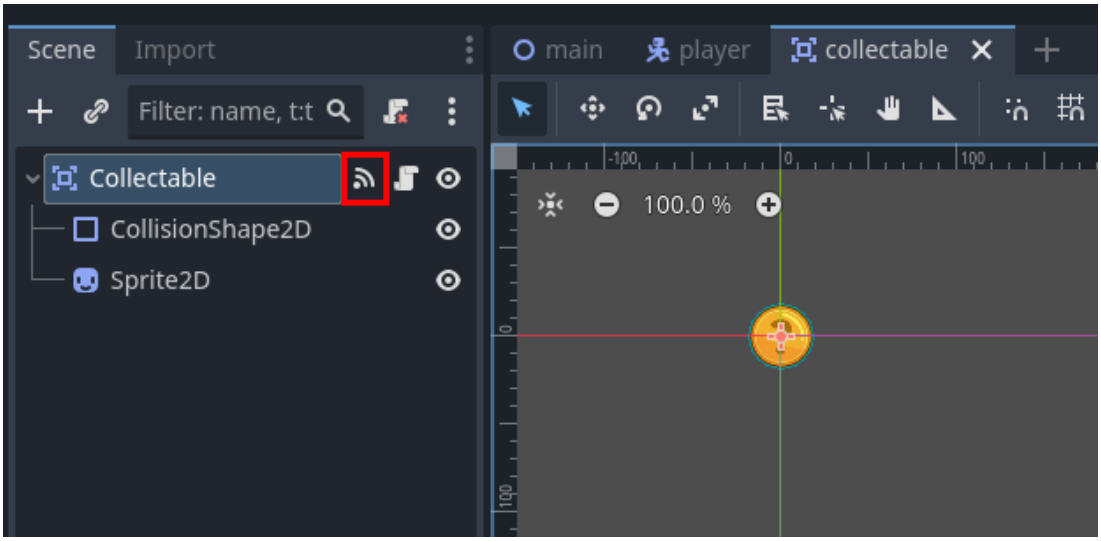
52 This creates a new **subtree** node, just like the **Player**. Open the scene by clicking the **scene** icon.



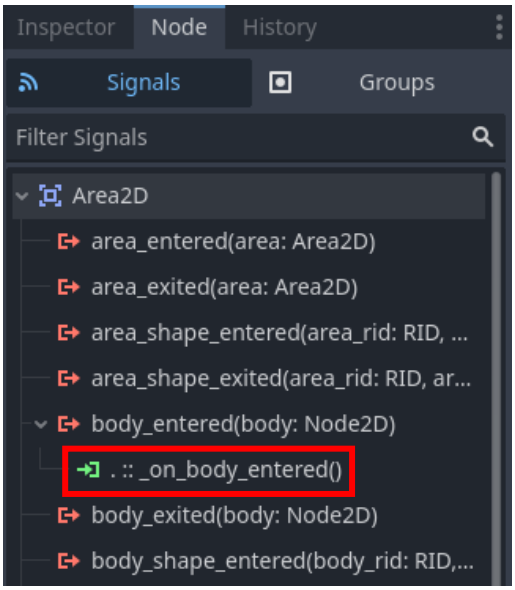
Note: The **collectable** scene already has a few things:

1. CollisionShape2D node
2. Sprite2D node
3. Attached Script
4. A signal icon

53 Click the signal icon next to the **Collectable** node. In the **Node** menu, find the **Area2D** section.



Notice the **on_body_entered** signal.



Signals are messages that nodes can send to each other when something specific happens to them. Think of them like **events** from IMPACT. For example, a signal could act like an **OnButtonEvent** which codes something to happen when a button is pressed.

What event from IMPACT do you think the **on_body_entered** signal could be?

Pro Tip:

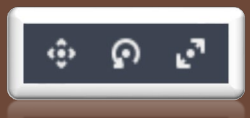
This signal emits when the Player's collision shape enters the Collectable's collision shape, so the code knows when to increase the score and delete the collectable. This signal acts just like an **OnOverlap** event from IMPACT!

54 Ensure **2D** is selected at the top of the editor, then go back to the **main scene**.

The new **Collectable** should appear in the middle of ScavengerHuntBG. Drag it closer to the Player to reposition it for testing.

Reminder:

At the top of the game window, these tools can be used to move, rotate, and resize objects. Hover over them to see their description!

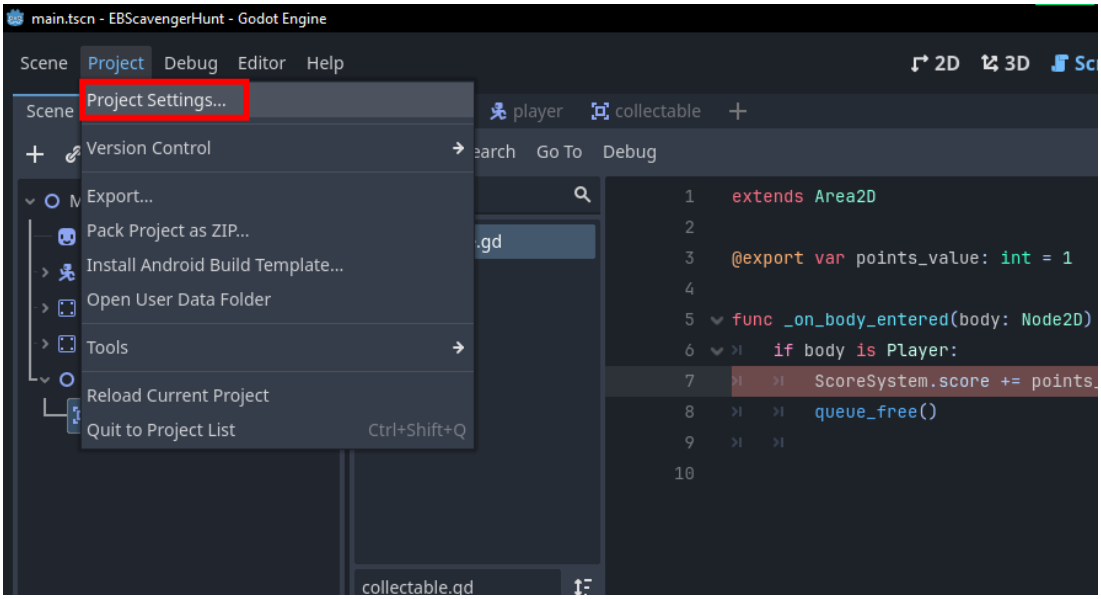


55 Playtest the game. What happens?

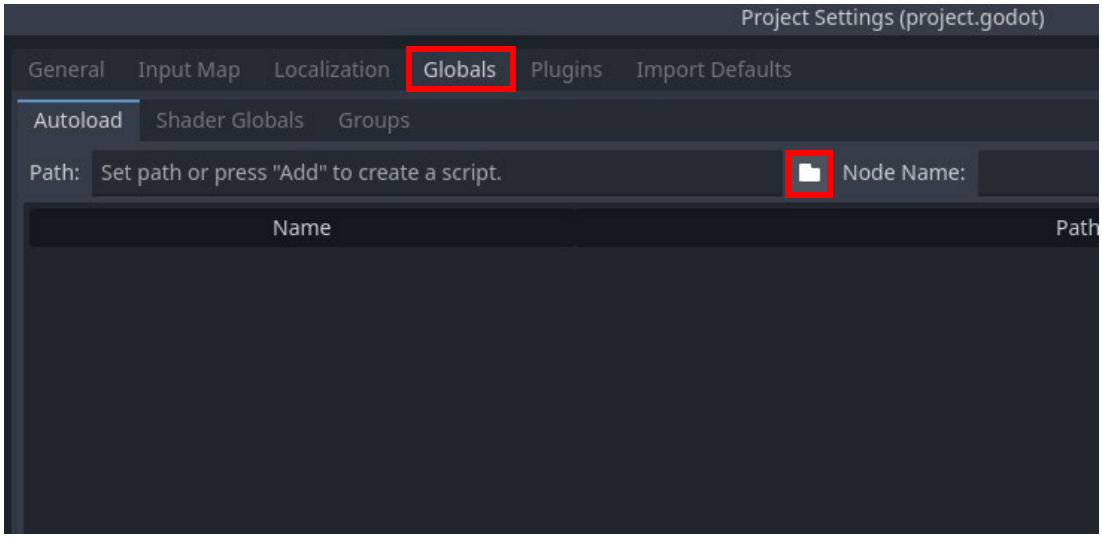
There's an **error**! This is because the script expects to see a **ScoreSystem** script to increase its score.

Add a **Global Script** to resolve the error**.** Global scripts live *outside* of individual scenes and can be accessed by all other scripts in the game, allowing them to keep track of the player's score, for example.

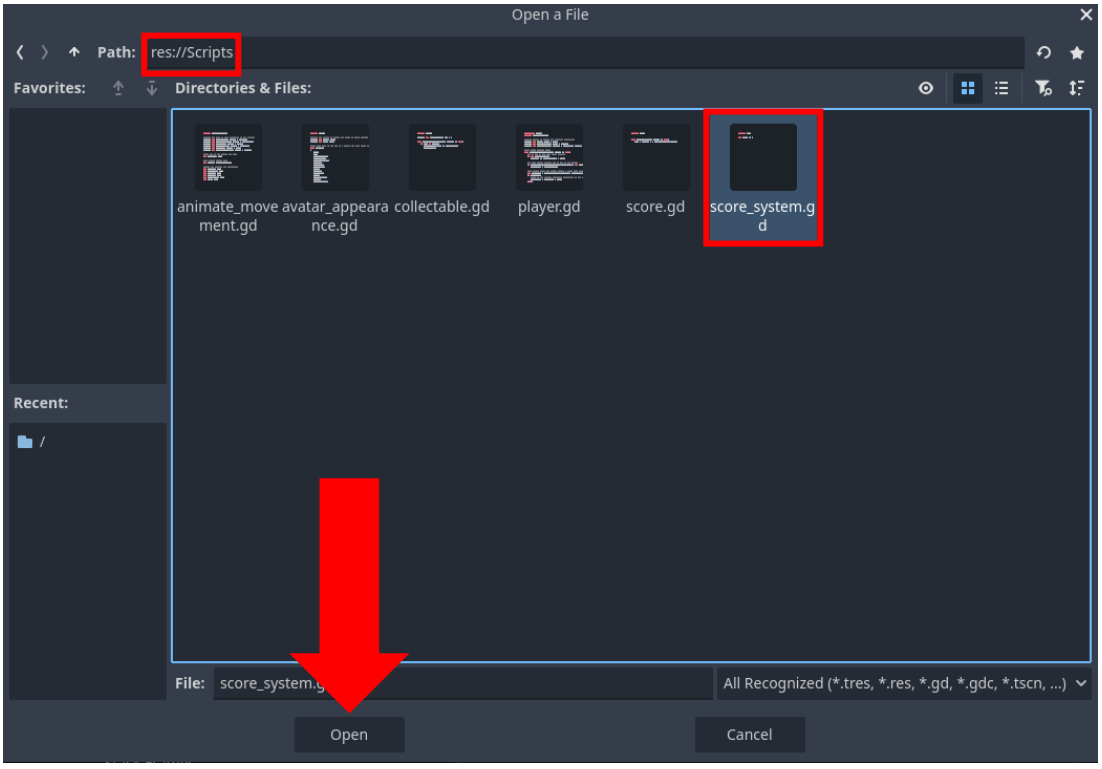
Click on **Project** and then **Project Settings**.



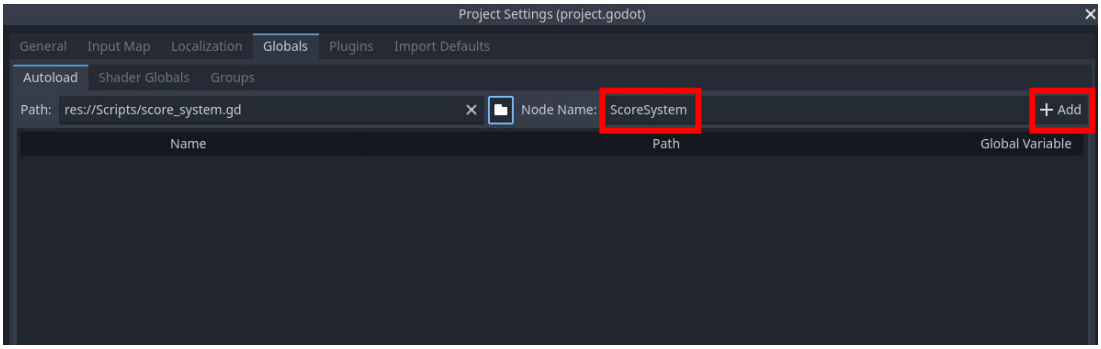
56 Navigate to **Globals** and ensure the **Autoload** section is open. Click the **folder** icon.



Open the **Scripts** folder and select **score_system.gd**. Once the script is selected, click **Open** at the bottom.

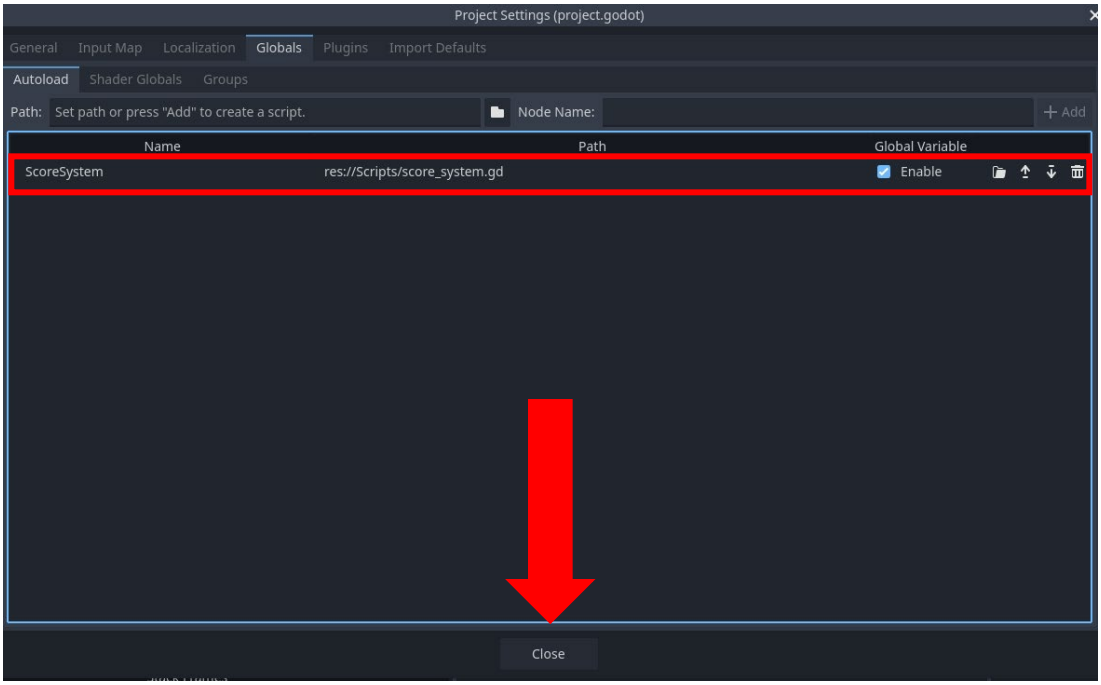


57 Check that the Node Name is **ScoreSystem**. Click **+Add** on the right to add the **Global Script**.

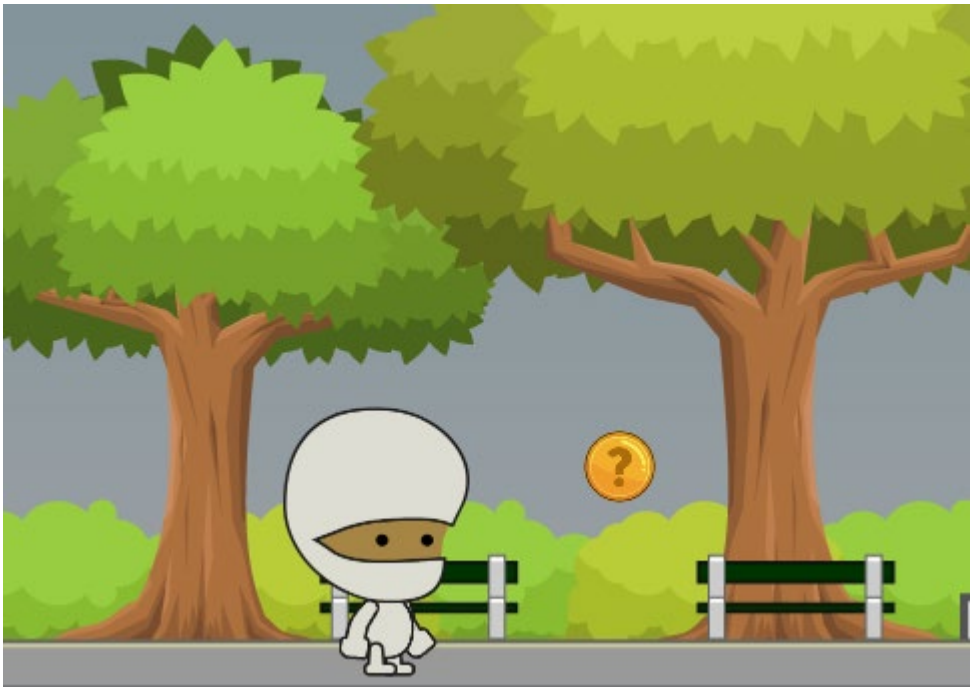


58 The **Project Settings** window should match the image. If it doesn't, review the previous steps.

Click **Close** to exit the **Project Settings** window.



59 **Playtest** the game. The **Collectable** should now **disappear** when the Player collides with it.

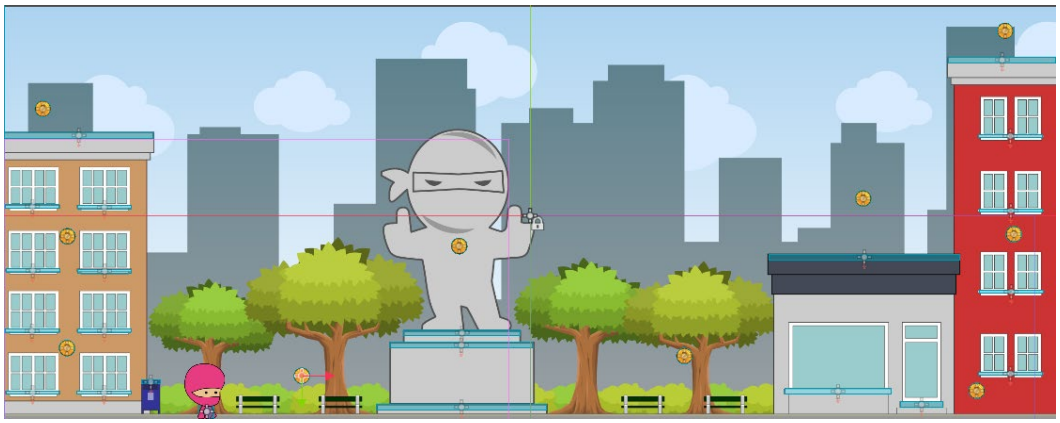


Pause for **Sensei Stop #7**!

Check in with a Code Sensei before moving on. Have the Code Sensei **playtest** the game to make sure the **Collectable disappears** after the Player collides with it.

Reminder: Press **CTRL + S** to save your work!

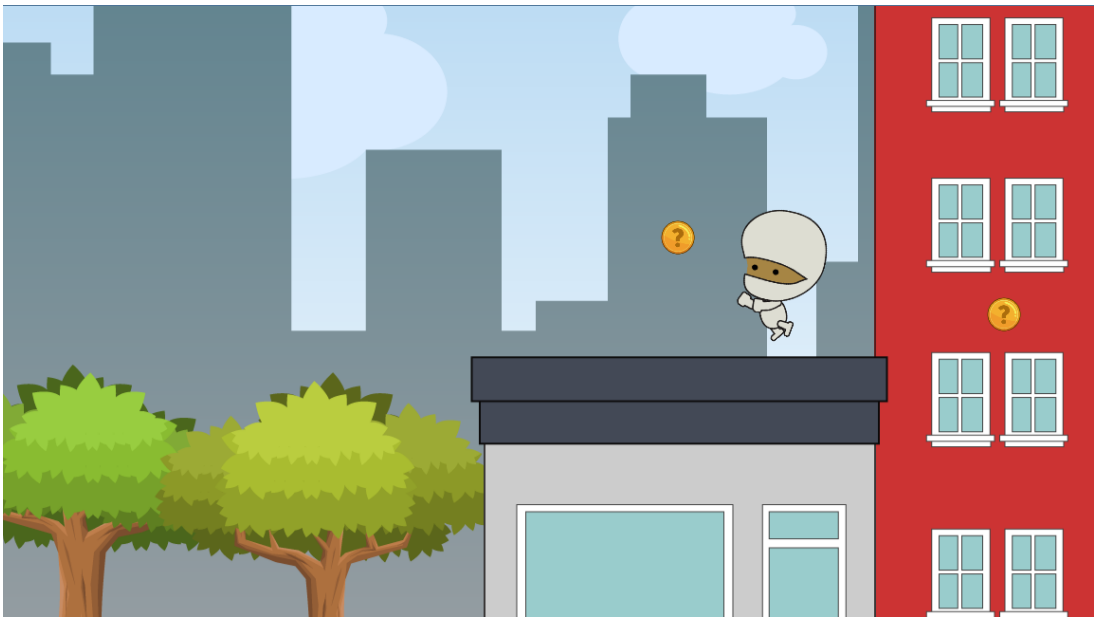
60 Add at least **10** collectables and reposition them throughout the scene to places where they might be challenging to reach.



Reminder:

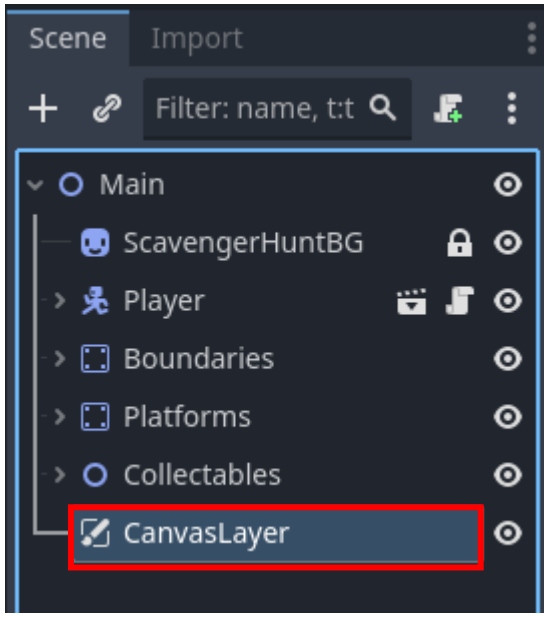
To quickly create duplicates, press **CTRL + D**. Undo mistakes by pressing **CTRL + Z**. Use the **move** icon at the top of the editor to move the player around.

61 Playtest the game and try to collect all the collectables as fast as possible!



62 Hmmm, there's currently no way to see the player's score. Let's fix this by adding a **User Interface (UI).** User interfaces are used to implement score tracking, a health bar, or other items to display to the user.

Add a **CanvasLayer** node as a child node to the **Main root.**

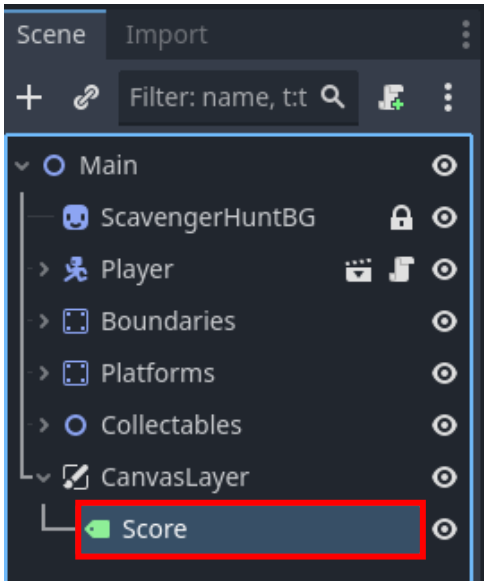


New Concept: CanvasLayer

A **CanvasLayer** node makes sure that the UI stays in one place on the screen. Imagine a layer at the very top of the game window that's just for the UI; it receives information from the game, but the elements of the game cannot touch or move the UI.

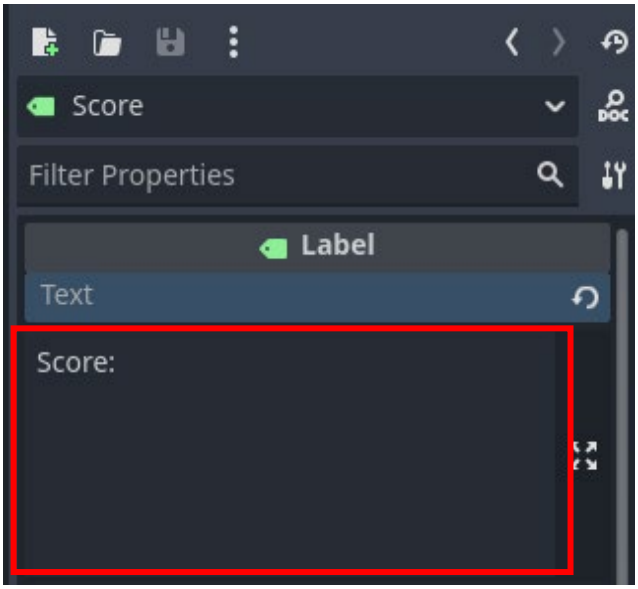
63 Add the score label.

Add a **Label** node as a child node to **CanvasLayer,** then rename it **Score.**

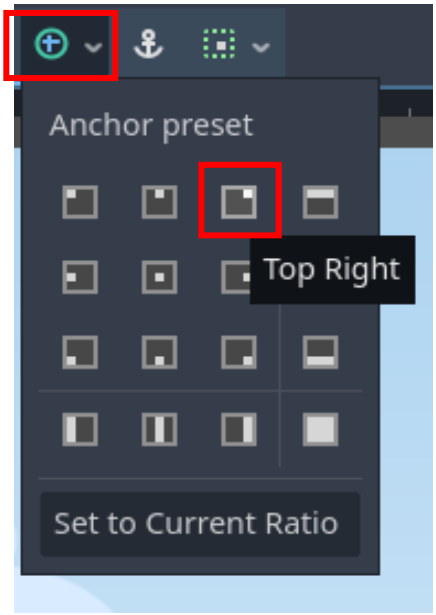


Note: A **Label** is a UI node used to display **text** on the screen.

64 In the **Inspector** under **Text,** type "**Score:**". This is just a placeholder to see how the label will display the score on the screen, like **"Score: 10".**



65 At the top center of the editor, select the **Anchor Preset icon.** Select an anchor preset that will position the score anywhere at the *top* of the screen. The **anchor** will keep the score in place, like an anchor to a ship!



Pro Tip:

Hover over the anchor presets to see what position the text will be placed in.

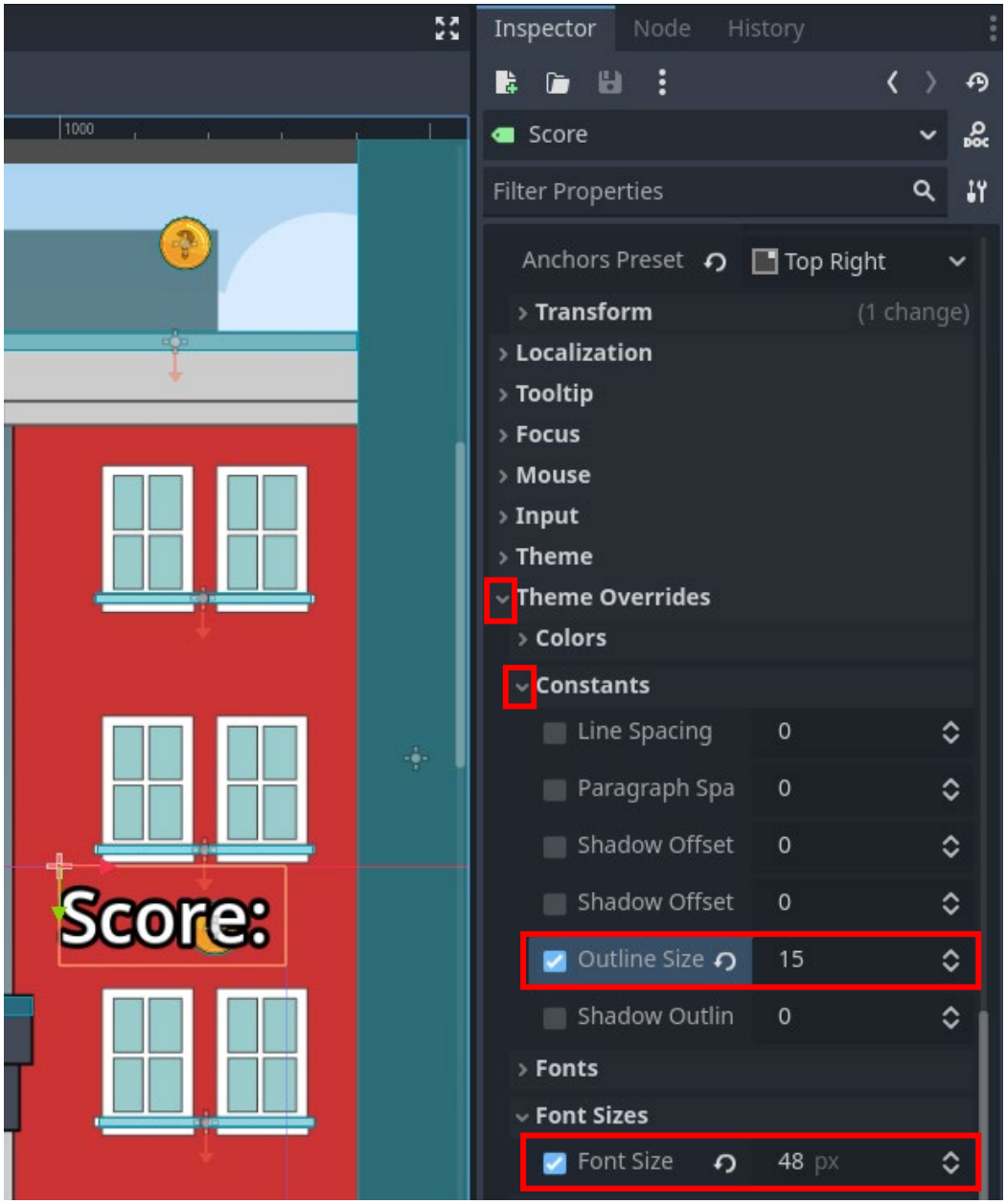
66 Playtest the project and notice a tiny **Score** label at the top of the screen.

Move the player around and notice how the score stays in the same position.

67 Increase the score's visibility on screen.

In **Inspector** for the **Score,** scroll down and open the **Theme Overrides** drop-down menu**.** Open the **Constants** drop-down menu and change **Outline Size** to a value between **5-20**.

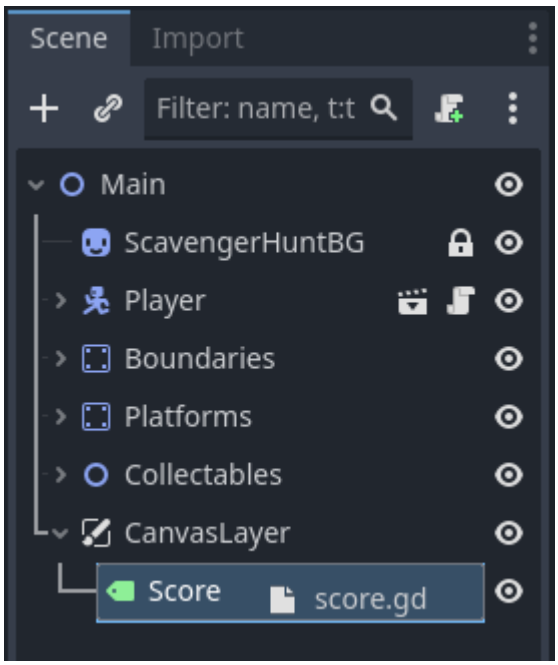
Open **Font Sizes** and use a value between **40-60** to adjust the size.



68 Playtest the project again. Try to collect some collectables; what do you notice about the score?

69 Uh oh, the score is not updating as the Player collects the collectables! That's because the **score script** needs to be attached.

In **FileSystem**, locate the score script **score.gd.** Drag **score.gd** onto the **Score** node. Locate the **script** icon next to the **Score** node to confirm the script has been added correctly.



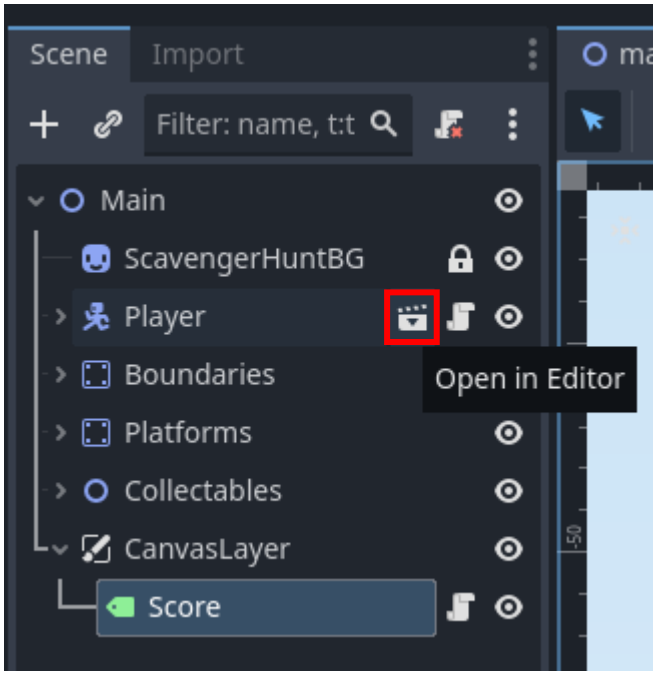
Pause for **Sensei Stop #8**!

Check in with a Code Sensei before moving on. Make sure that the **score updates** whenever a collectable is touched!

Reminder: Press **CTRL + S** to save your work!

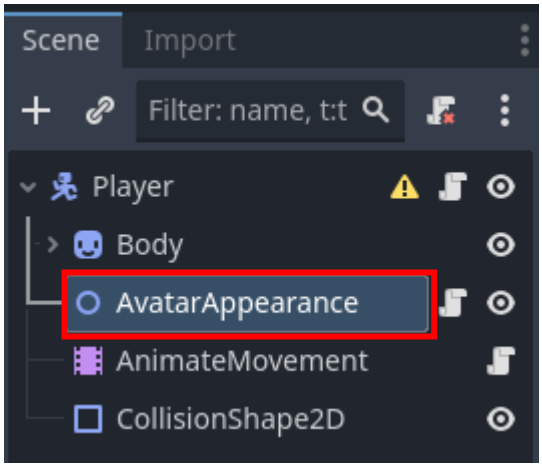
70 Time to customize the player!

Navigate to the **player** scene by clicking on the **scene** icon next to the Player node.



71 In the **player** scene, click on the **AvatarAppearance** node.

There is already a script attached that allows the player to easily be customized!



Reminder:

Select the arrow next to a parent node to hide its children.

72 In **Inspector**, explore the different ways to customize the player! After making a change, playtest the project to see how the player looks.

Try out different accessories and outfits!

Pause for **Sensei Stop #9!**

Congratulations on your first platformer game in Godot! Great job!

- What did you learn about Godot's hierarchy in the **Scene** menu?
- What did you enjoy most when creating this project?
- What was something you found difficult and why?

Reminder: Press **CTRL + S** to save your work and submit!