CODE NINJAS®

Silver Belt Ninja Guide

Activity 15: Amazing Ninja Worlds Part 3

Title Screens

UI, or User interfaces, are more than just information during gameplay. Menus, title screens, and any other 2D overlay within a game is classified as UI. Title screens are an important UI element; they serve as user's first impression of the game. The UI design sets a new user's expectations of what the game might be like. Returning users will find value in choosing which level to play, whether they are continuing the game or replaying their favorite parts. Title Screens also typically serve as a navigation menu so the player can select the portion of the game they would like to play. Level selection is common in title screen interactivity.

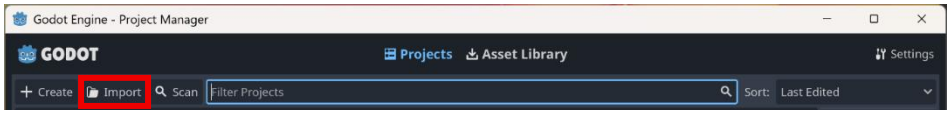
Think back to designing UI in lower belts of IMPACT. Consider the UI design limitations of MakeCode Arcade, which required utilizing tilemaps and sprites to create attractive, organized title screens and level selects. Conversely, Godot includes a host of built-in control nodes to make creating UI easier and more flexible.

The **HBoxContainer** is a control node that eases the process of creating visibly pleasing and easily understood UI. This node houses other control nodes and includes many settings to organize the nodes evenly in a horizontal arrangement. This alleviates the need to individually place each constituent control node in the UI, instead automatically arranging them in an organized manner.

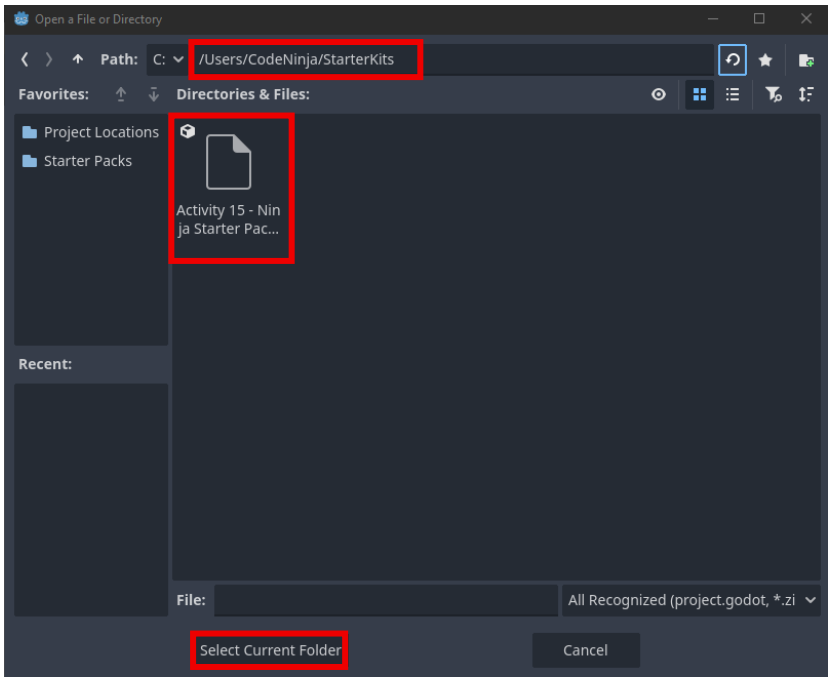
Activity 15: Amazing Ninja Worlds Part 3

In this project, you will create a new UI scene that contains title screen UI. You will script the title screen UI to show up at the beginning of the game, and the logic that will load the correct level when it is selected.

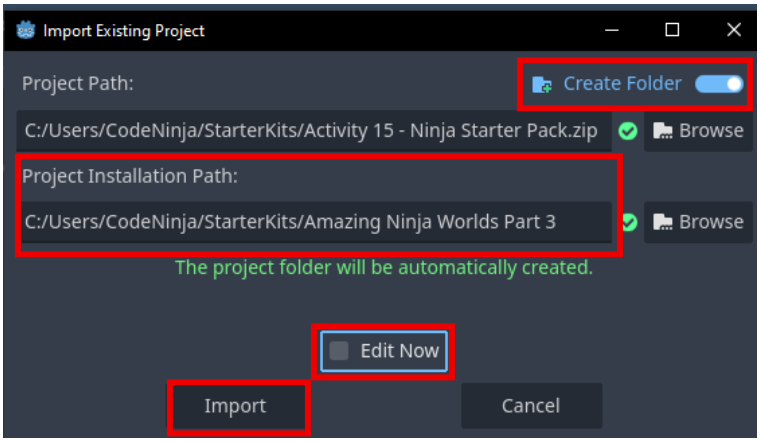
1 Open Godot and click **Import.**



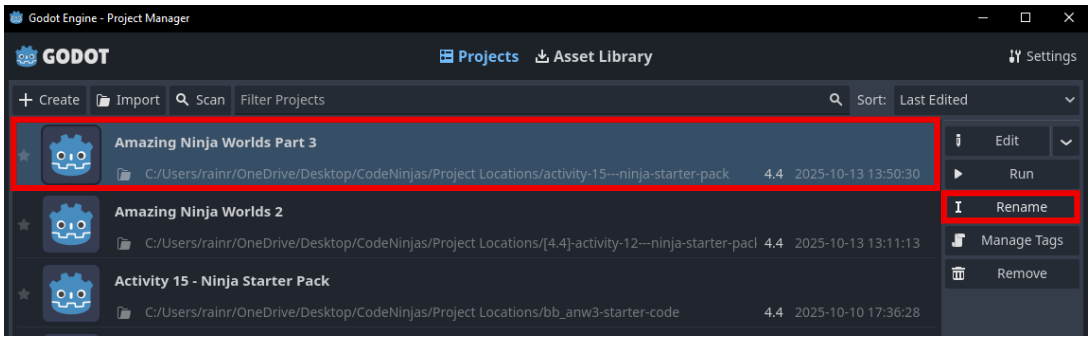
In the File Directory, navigate to the correct file path. Select **SB Activity 15 – Ninja Starter Pack.zip** and click **Open**.



2 Update the **Project Installation Path** and make sure **Create Folder** is enabled. **Uncheck Edit Now** and click **Import**.

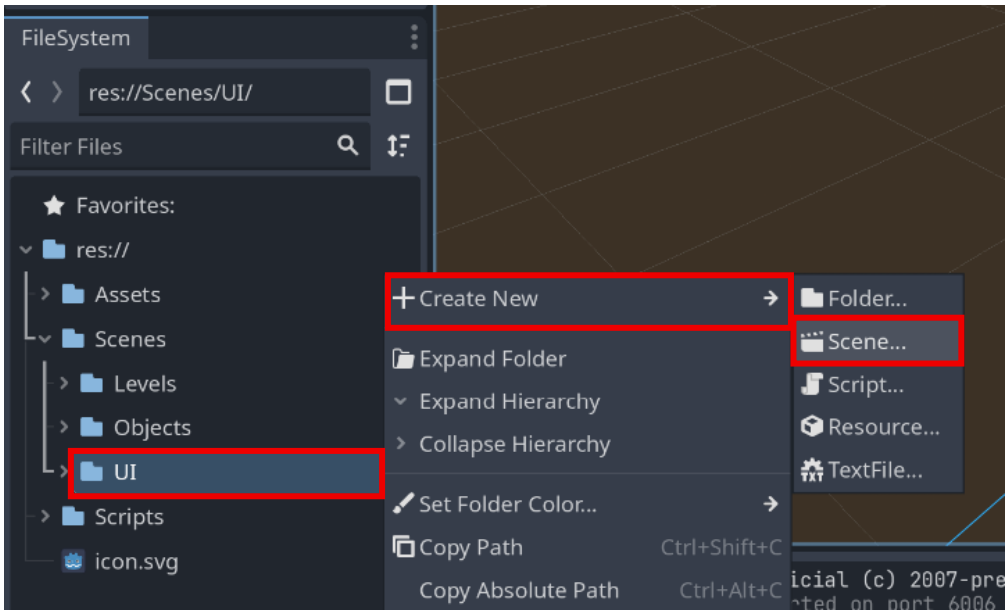


3 The project will appear at the top. Click on the project and select **Rename** on the right. Update the Project Name to **[YourInitials]AmazingNinjaWorlds3**.

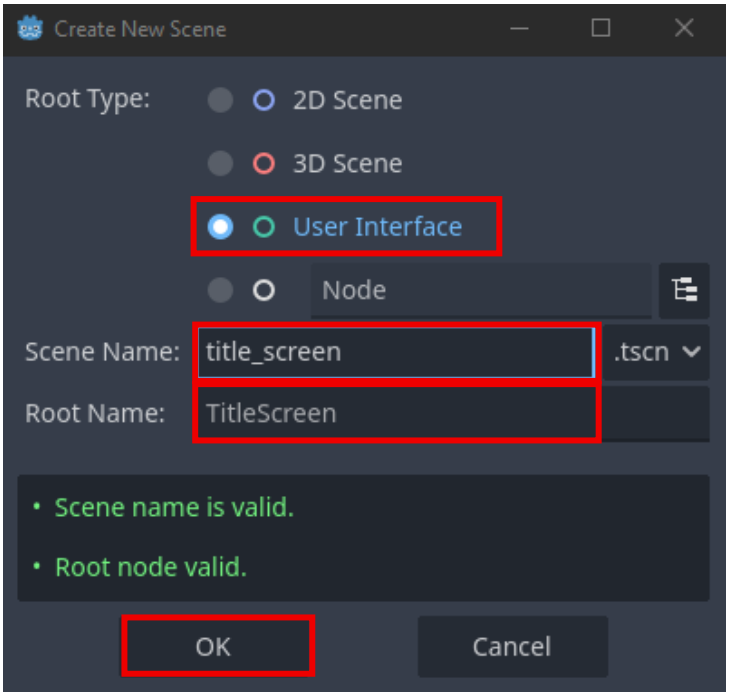


Once renamed, select the project and click **Edit** to open the starter code.

4 In **FileSystem**, under **Scenes**, right click the **UI** folder and select **Create New > Scene.**

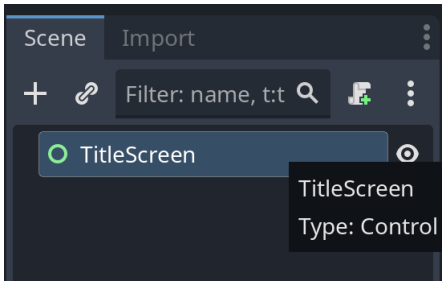


5 In the Create New Scene window, select the **User Interface** root type. Then, set the scene name to **title_screen**. The root name should update automatically to **TitleScreen**.



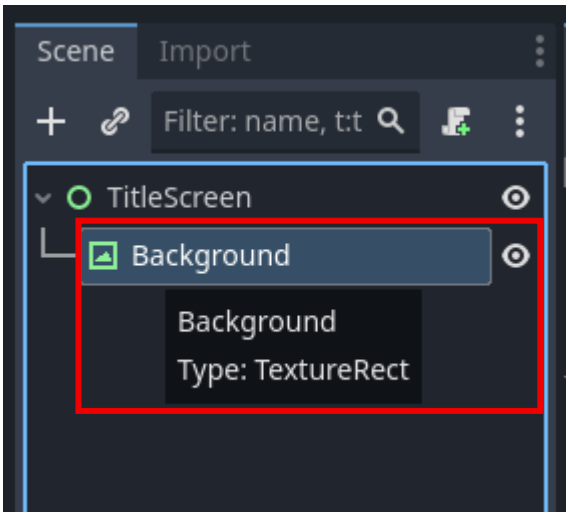
Click **OK**. The **title_screen.tscn** scene should automatically open in the 2D workspace.

6 Notice that the root node for the scene is a **Control** node.

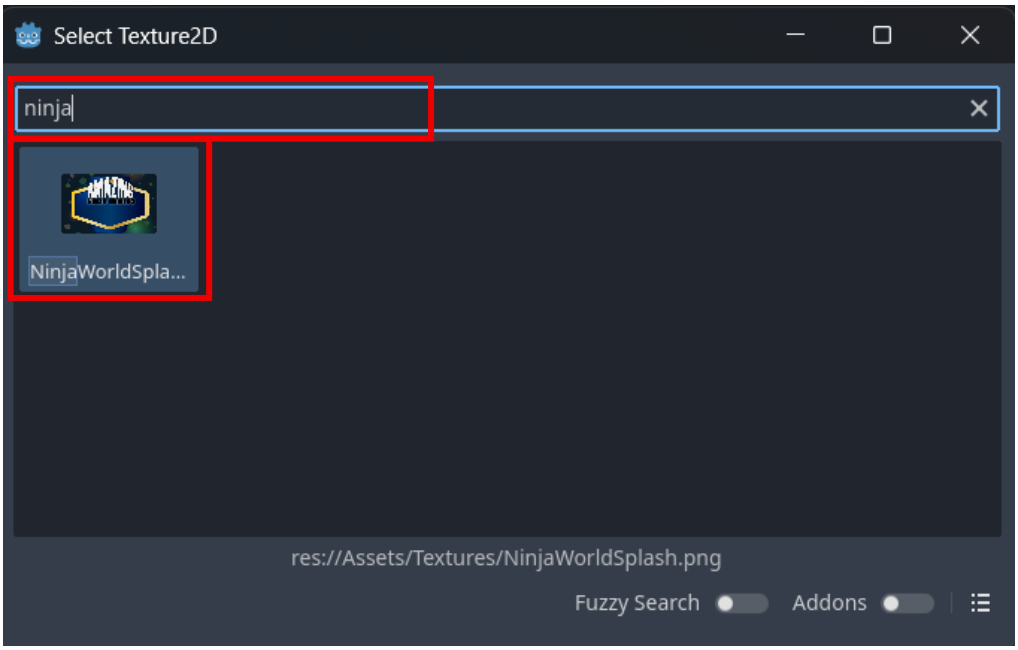


Control nodes should be used as the root of a scene when the UI will *not* overlap with game play. Because this scene will contain only UI elements, including the title and level select buttons, a control node is preferred over a CanvasLayer.

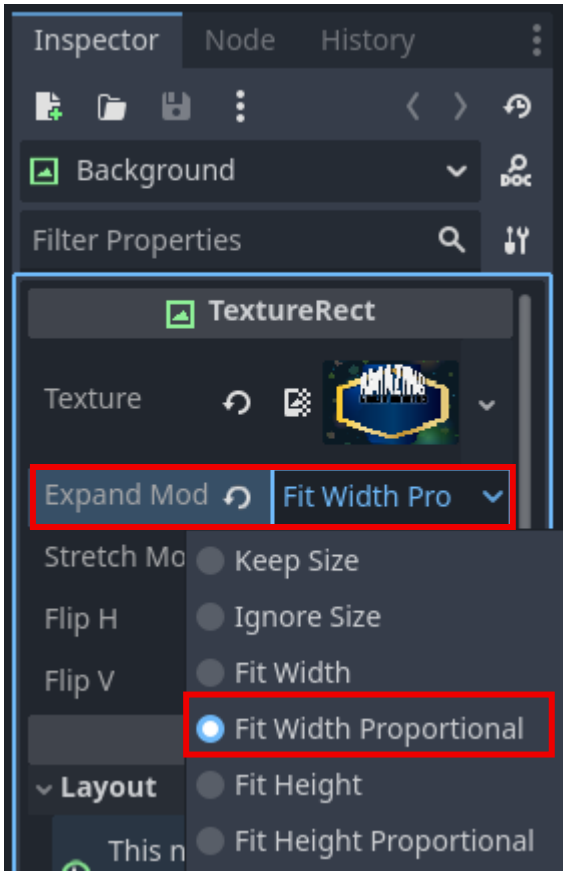
7 In **Scene**, add a **TextureRect** as a child to **TitleScreen** and rename the node **Background**.



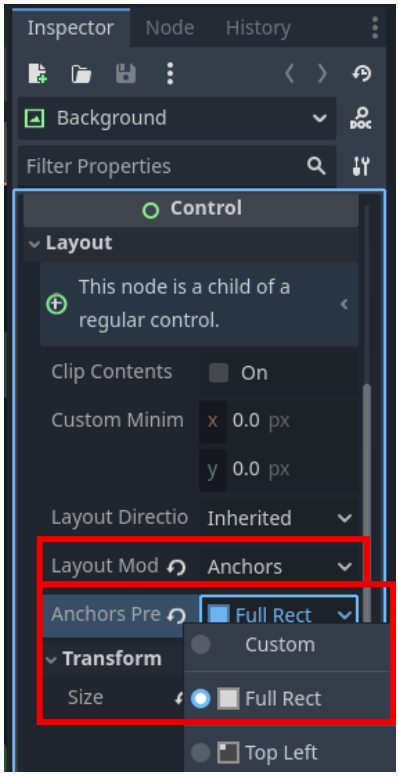
In the **Inspector** for **Background**, use **Quick Load** to set the Texture by searching for the **NinjaWorldSplash.png**.



8 In the Inspector for Background, set the **Expand Mode** property to **Fit Width Proportional**. Notice the change in the texture's size.



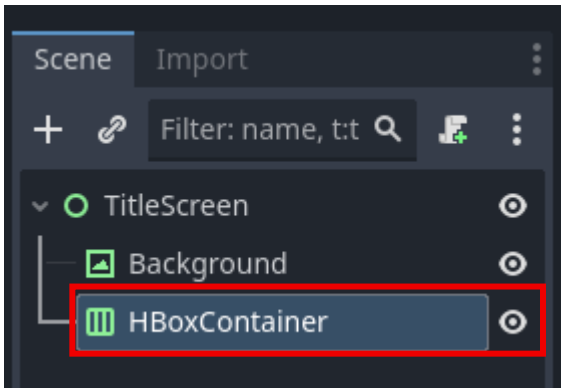
9 Under Layout, set the **Layout Mode** property to **Anchors** and set **Anchors Preset** to **Full Rect**.



In the **2D workspace**, notice how the background image takes up the entire screen.

10 The title screen will need a button for each of the three levels. These can be organized with an **HBoxContainer**.

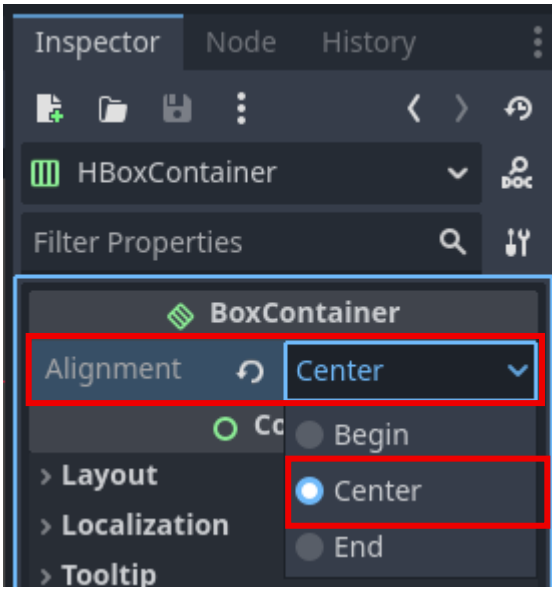
In **Scene**, add an **HBoxContainer** as a child to **TitleScreen.**



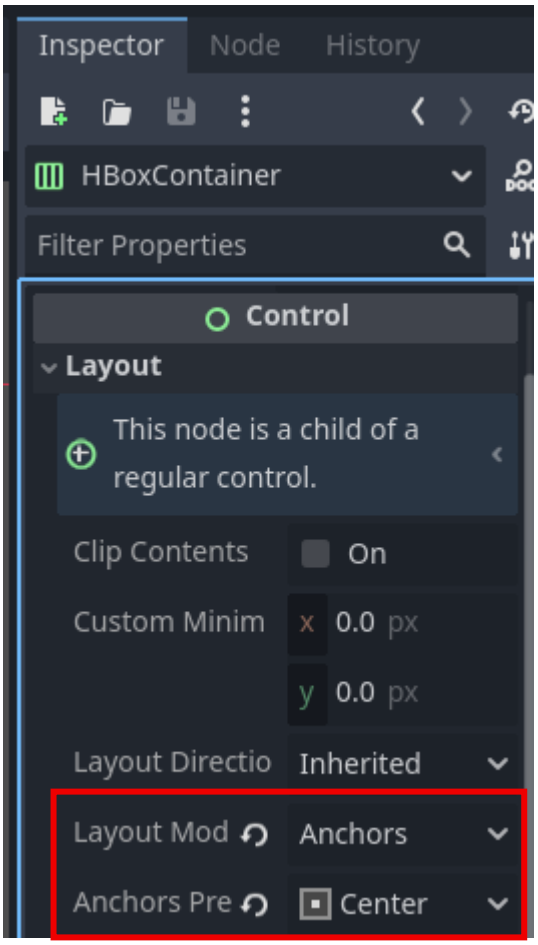
Reminder:

An **HBoxContainer** node is a type of control node that will arrange its child control nodes horizontally. This helps organize control nodes neatly.

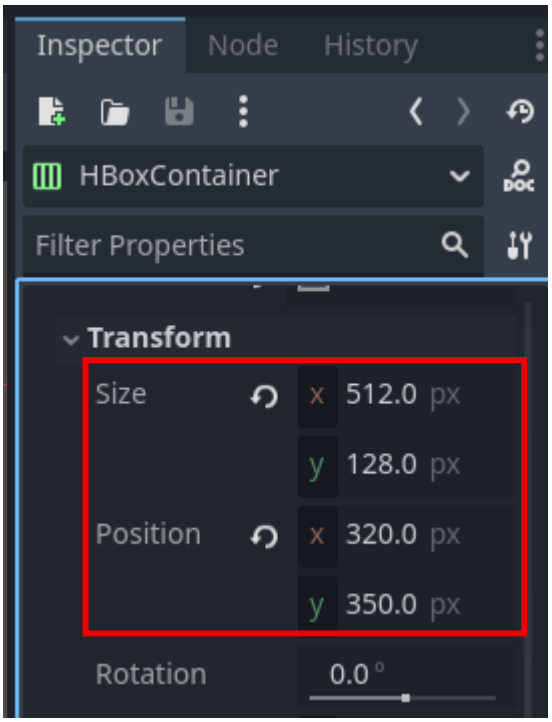
11 In the **Inspector** for **HBoxContainer**, set the **Alignment** to **Center**.



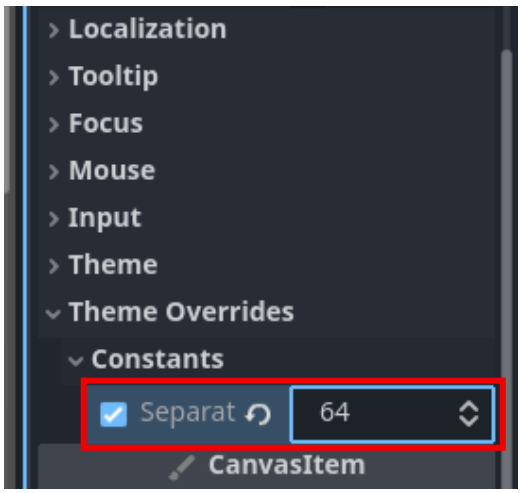
12 Under **Layout**, set the **Layout Mode** to **Anchors** and set the **Anchors Preset** to **Center**.



13 Under Transform, set **Size x** to **512** and **Size y** to **128**, then set **Position x** to **320** and **Position y** to **350**.

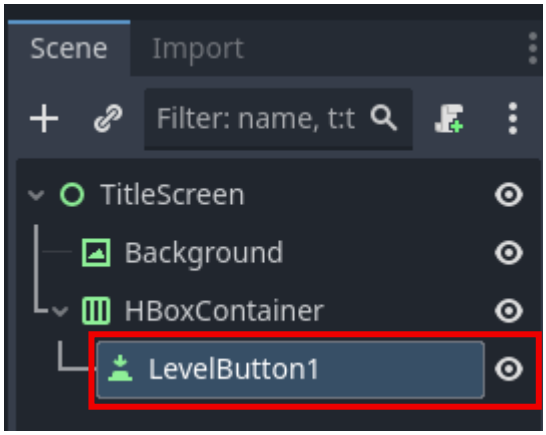


14 Under **Theme Overrides > Constants**, set **Separation** to **64** and ensure it is checked on. What might changing the value of this setting do?

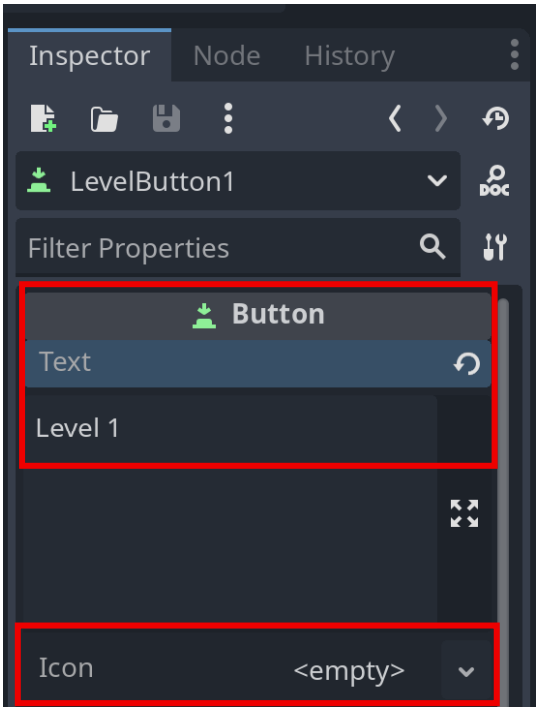


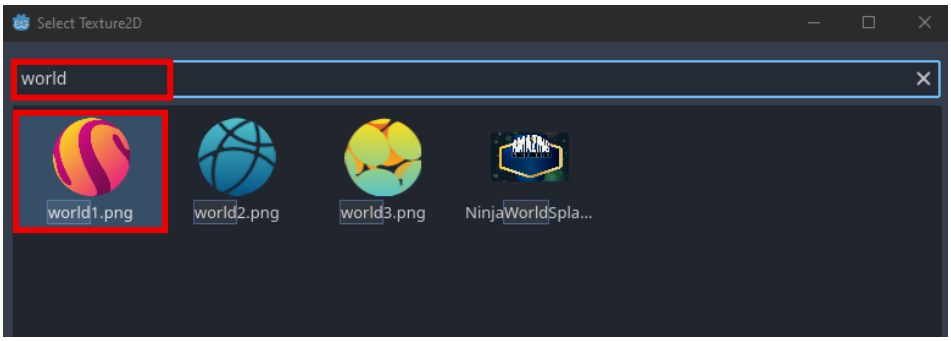
Separation is the space between the HBoxContainer's elements in pixels.

15 In **Scene**, add a **Button** as a child to **HBoxContainer** and rename it to **LevelButton1**.

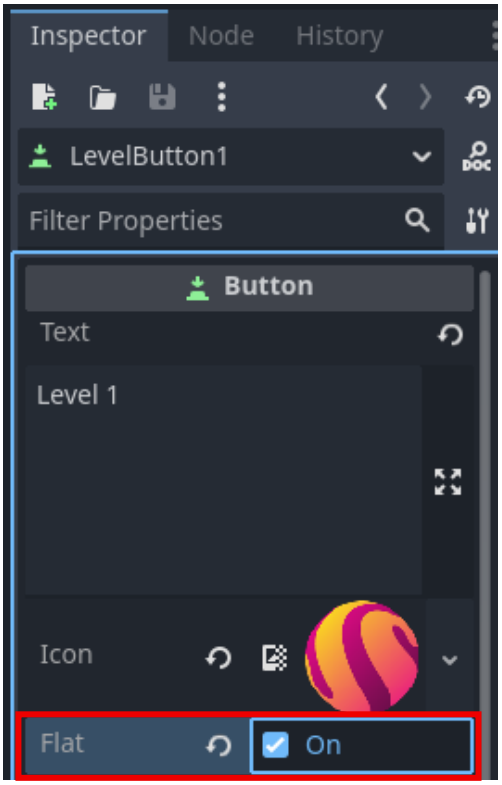


16 In the **Inspector** for **LevelButton1**, set **Text** to **Level 1**, then use **Quick Load** to search for and set the **Icon** to **world1.png**.

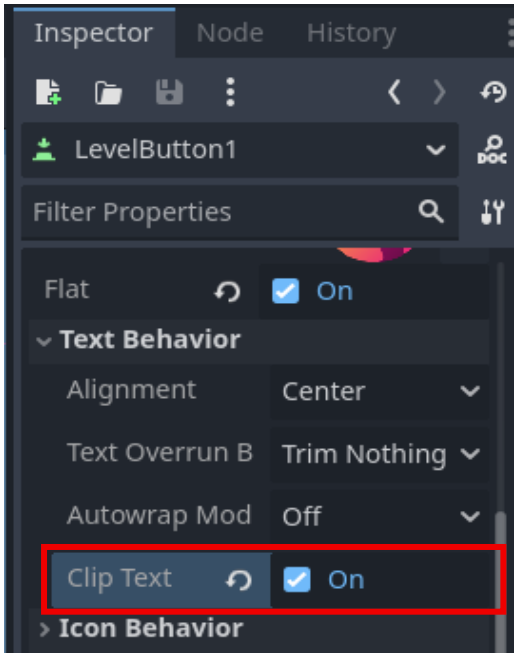




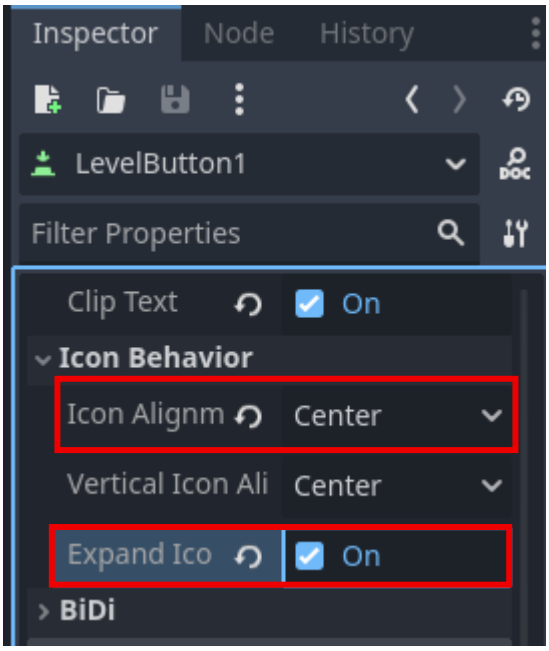
17 Check the **Flat** property **On**. Notice what happens when the **Flat** property is toggled. What might this setting do?



18 Under **Text Behavior**, check the **Clip Text** property **On**. This will cut off the button's text if the text becomes too large to fit the button.

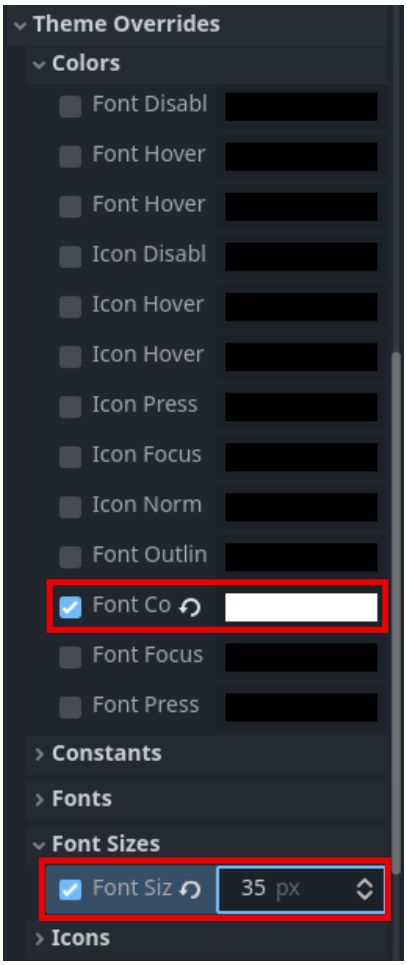


19 Under **Icon Behavior**, set **Icon Alignment** to **Center**. Check the **Expand Icon** property **On**.



20 Under **Layout**, set **Custom Minimum Size x** and **y** both to **128**.

21 Under **Theme Overrides > Colors**, check the **Font Color** property **On** and set the color to **White** using the hex value **#FFFFFF.** Then, set **Font Size** to **35** ensure it is checked **On**.



The title screen should look like the image below.

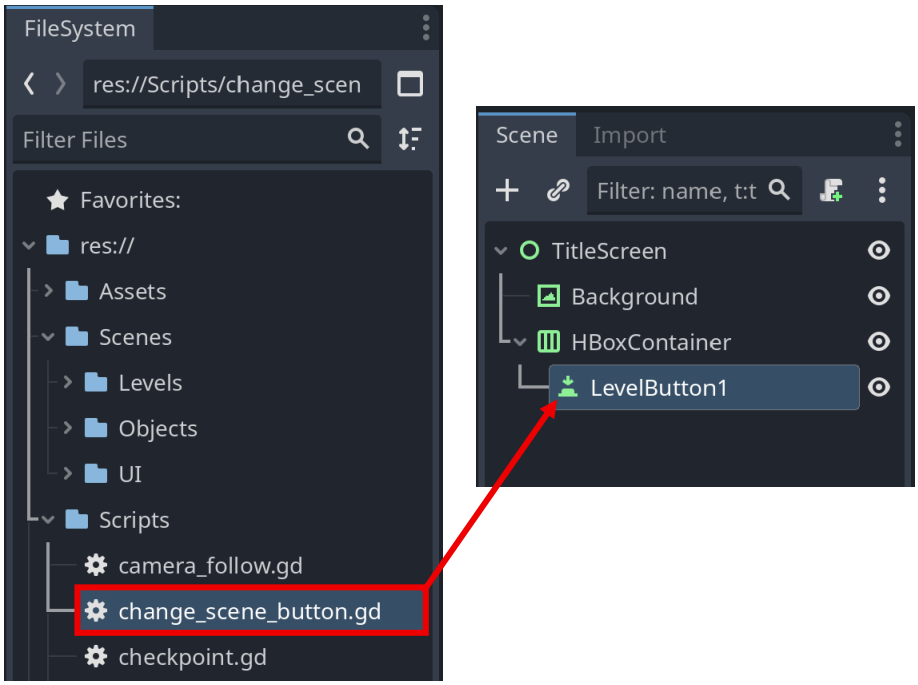
Pause for **Sensei Stop #1**!

Check with a Code Sensei and confirm the title screen scene, its nodes, and the LevelButton1's properties are properly set up before continuing.

Reminder: Save your work!

22

In **FileSystem**, locate the **change_scene_button.gd** script in the **Scripts** folder and attach the script to the **LevelButton1** node.



Once attached, click on the **script icon** next to the **LevelButton1** node to open the script.

23 The button needs to be coded to open the correct level when pressed.

Under **TODO 1** in **change_scene_button.gd,** use the `@export_file()` keyword to declare a `level_path` variable of type `String`. Pass `"*.tscn"` as the argument to `@export_file()`.

The `@export_file()` keyword allows a path to a scene object to be stored as a string in the variable `level_path`.

```
2
3   # ---------------------
4   # TODO 1
5   # Create export_file variable named level_path
6   # ---------------------
7   @export_file("*.tscn") var level_path : String
8
```

24 Under **TODO 2**, define an `_on_pressed()` function. The function takes no parameters and returns `void`.

```
9
10  # --------------------
11  # TODO 2
12  # Create _on_pressed() function and defer call to change scene
13  # ---------------------
14  func _on_pressed() -> void:
15
```

25 Inside the `_on_pressed()` function, chain together calls to methods `get_tree()` and `call_deferred()`. Pass **`"change_scene_to_file"`** and `level_path` as the two arguments.

When `_on_pressed()` is called, the scene will change to the level specified by `level_path`.

```
9
10  # --------------------
11  # TODO 2
12  # Create _on_pressed() function and defer call to change scene
13  # --------------------
14  func _on_pressed() -> void:
15      get_tree().call_deferred("change_scene_to_file", level_path)
16
```

Reminder:

The `call_deferred()` method will run the function given as the first argument when the Godot engine has idle time. Idle time mainly occurs at the end of physics and process frames.

26 Under **TODO 3**, use the code completion to define the `_ready()` method.

```
17
18  # --------------------
19  # TODO 3
20  # Create _ready() function and connect button to _on_pressed
21  # --------------------
22  func _ready() -> void:
23
```

27

Inside the `_ready()` method, chain together a reference to `pressed` and the `connect()` method with the argument `_on_pressed`.

This tells the button attached to the script to call the `_on_pressed()` function when pressed.

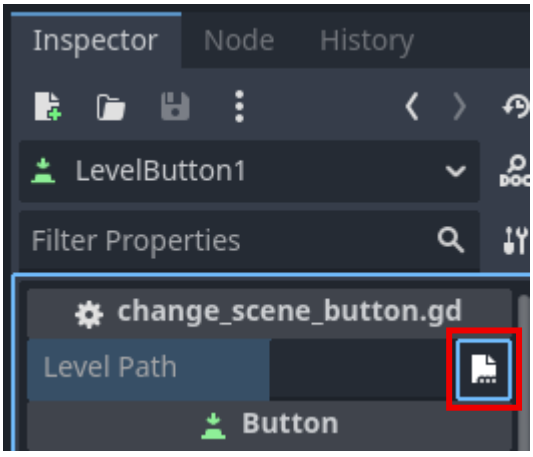
```
17
18  # --------------------
19  # TODO 3
20  # Create _ready() function and connect button to _on_pressed
21  # --------------------
22  func _ready() -> void:
23      pressed.connect(_on_pressed)
```

Reminder:

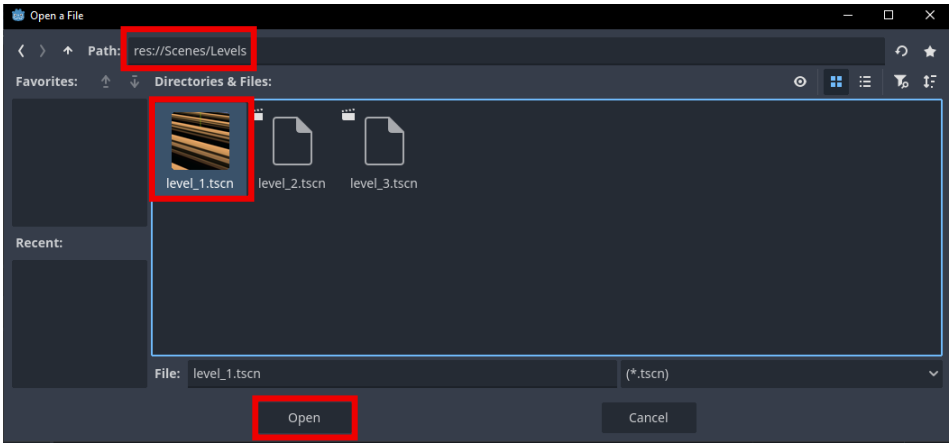
The argument given to the **`connect()`** method must be a reference to a function, not a call. This means that the parentheses **()** should not be included on the end of the function name.

28

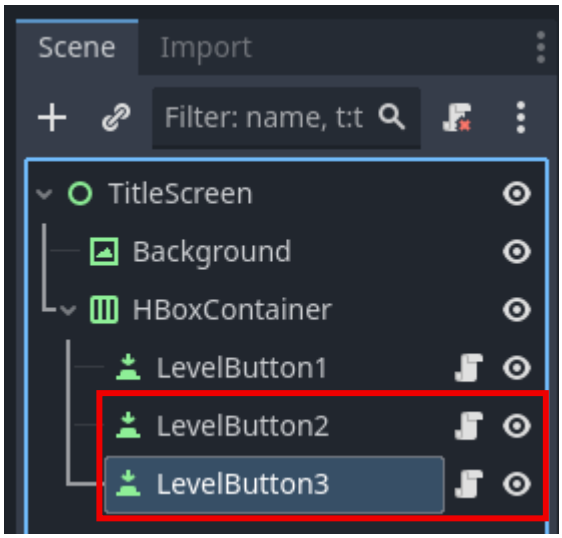
At the top of the **Inspector** for **LevelButton1**, click the file icon to the right of the **Level Path** property.



29 In the **Open a File** window, navigate to **Scenes > Levels.** Select the **level_1.tscn** and click **Open.**



30 In **Scene**, duplicate the **LevelButton1** node twice, creating 3 total buttons.

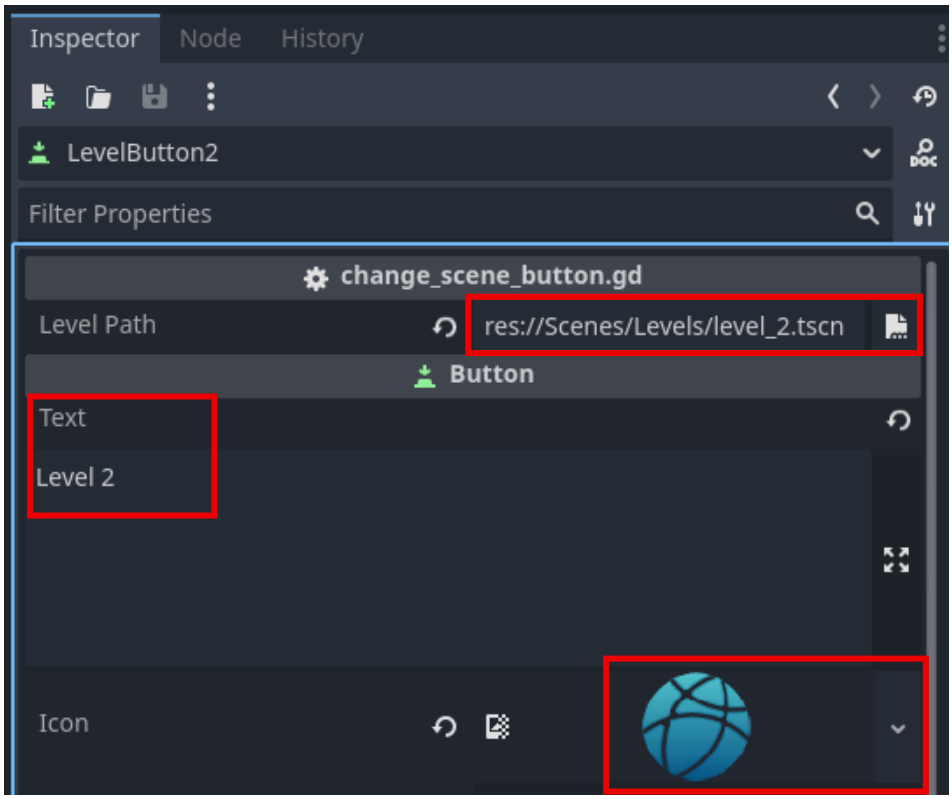


Pro Tip:

Press **Ctrl + D** on the keyboard while a node is selected to quickly duplicate it at the same level in the scene's hierarchy.

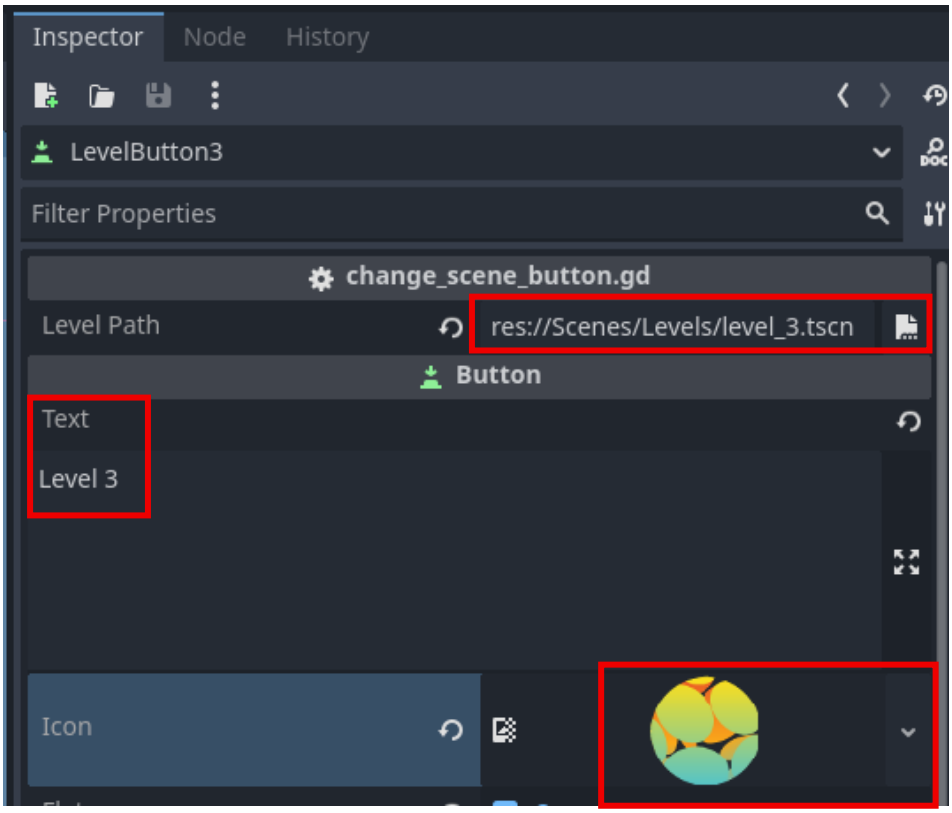
31 In the Inspector for LevelButton2:

- Set the **Level Path** to reference **level_2.tscn**.
- Change the **Text** to **Level 2**.
- Use Quick Load to set the **Icon** to **world2.png.**



32 In the Inspector for LevelButton3:

- Set the **Level Path** to reference **level_3.tscn**.
- Change the **Text** to **Level 3**.
- Use Quick Load to set the **Icon** to **world3.png.**



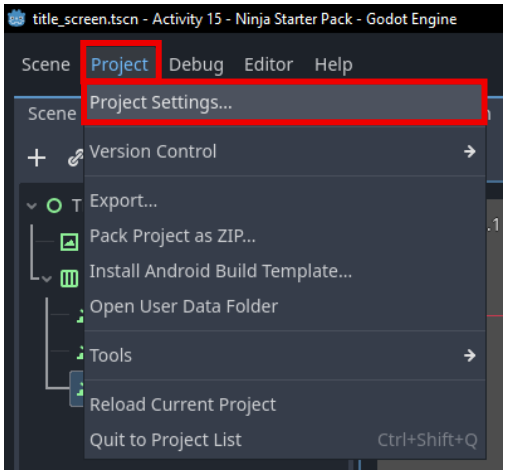
In the 2D workspace, notice that the **HBoxContainer** keeps the 3 level buttons evenly spaced.

Pause for **Sensei Stop #2**!

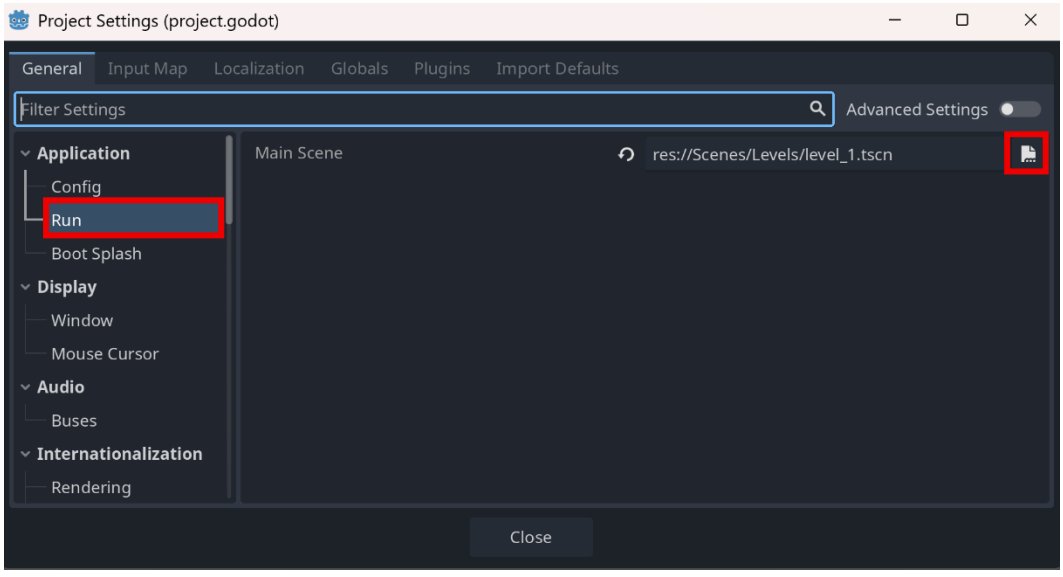
Check with a Code Sensei and confirm the LevelButton nodes' values are properly set up before continuing.

Reminder: Save your work!

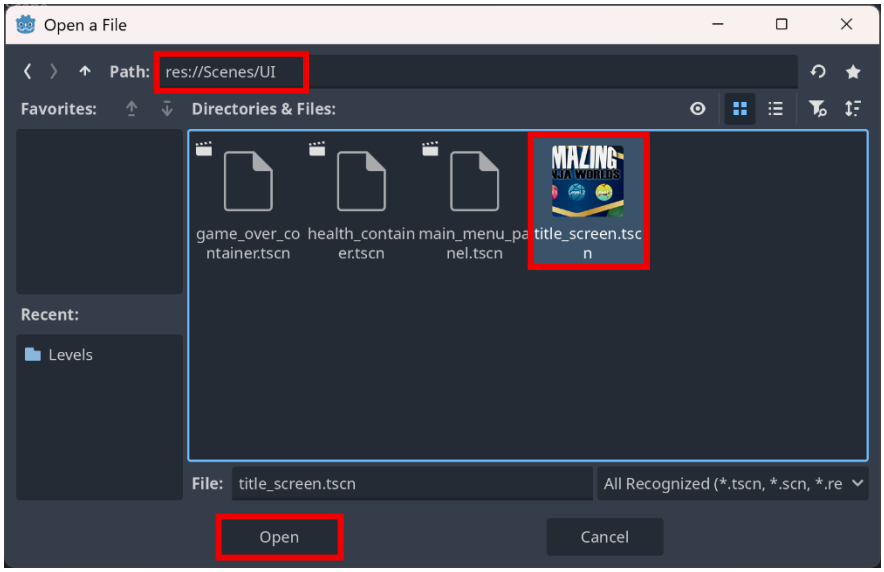
33 The project's main scene needs to be set to the title screen. At the top of the editor, click **Project** and select **Project Settings.**



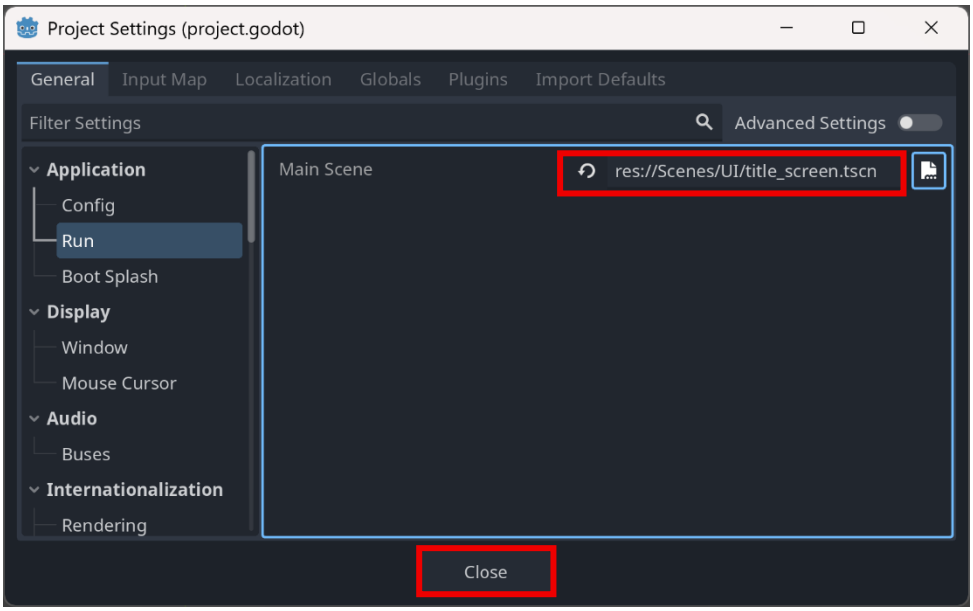
34 In the **Project Settings** window, under the **General** tab, navigate to **Application > Run.** Notice that the **Main Scene** is currently set to **level_1.tscn**. Click the **file icon** to the right of **Main Scene** to change it.



35 In the **Open a File** window, update the **Path** to **res://Scenes/UI**, then select **title_screen.tscn** and click **Open**.



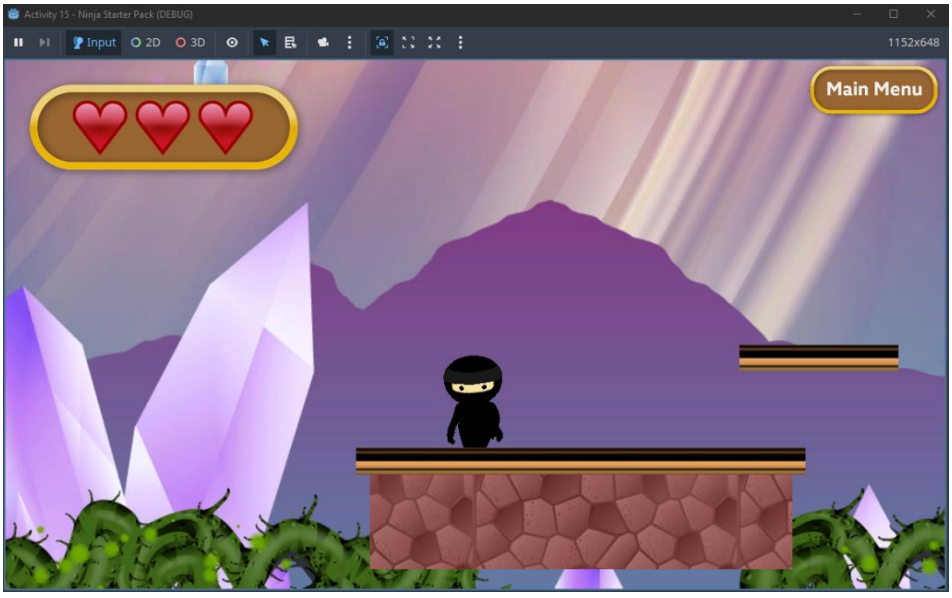
Check that the correct scene was chosen, then close out of the project settings window.



36 Playtest the game.

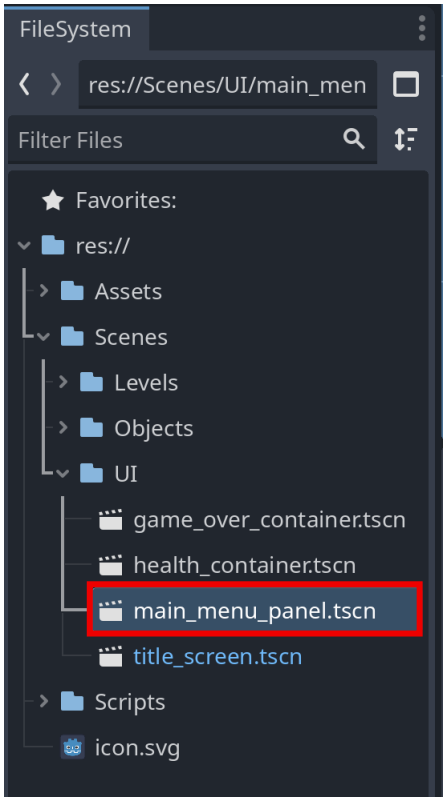
The game should load into the title screen UI. When one of the level buttons is clicked on, the correct level should load and become playable.

When inside a level, click on the Main Menu button. When the Main Menu button is pressed, what happens?



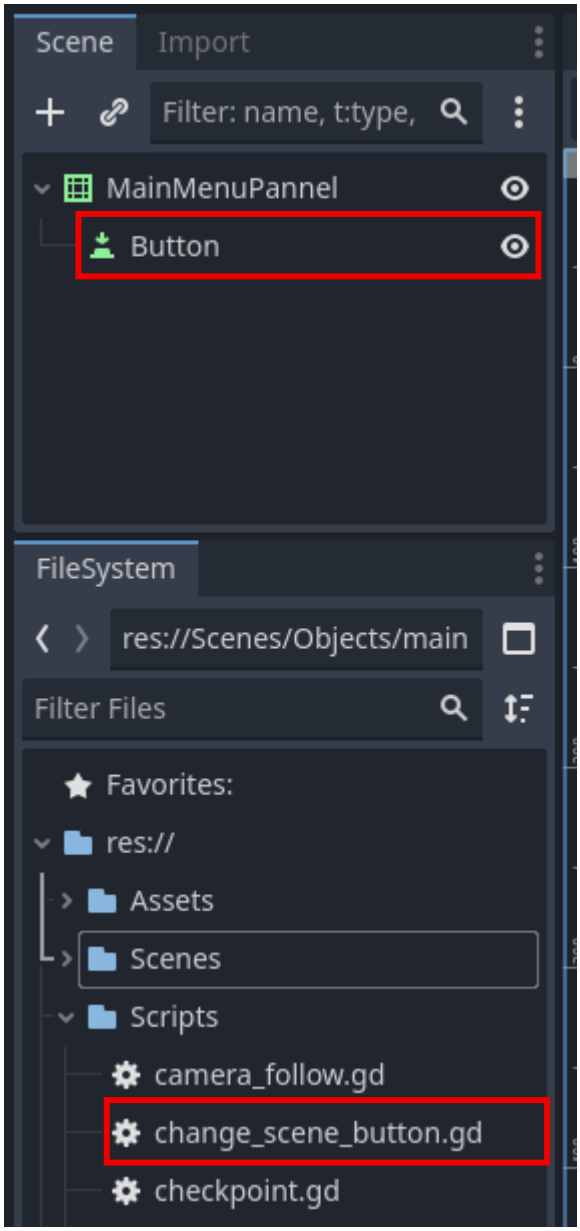
37 The game does not return to the title screen when the Main Menu button is pressed!

In **FileSystem**, navigate to **Scenes > UI** and open the **main_menu_panel.tscn** scene.

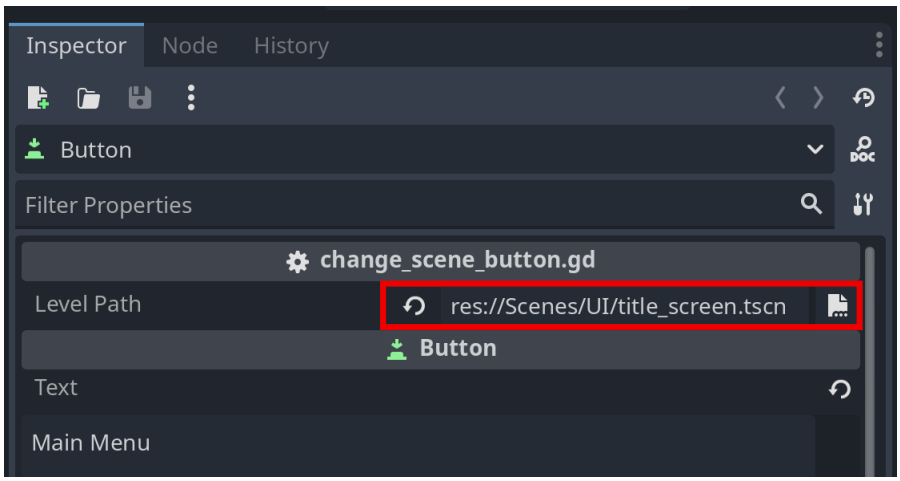


38 Notice there is no script attached to the button in the main_menu_panel.tscn scene.

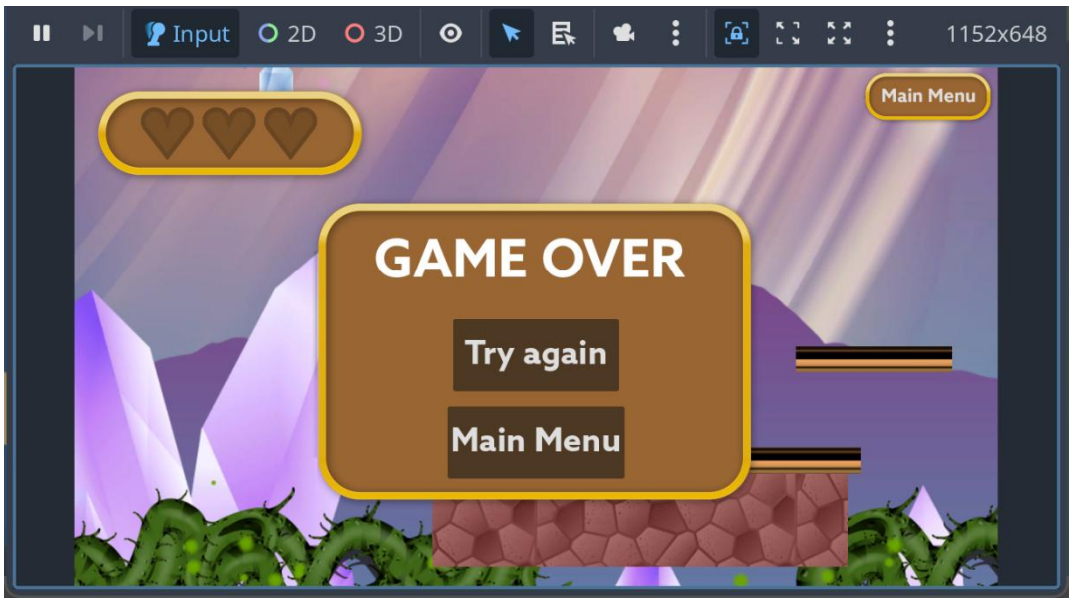
In **FileSystem**, under **Scripts**, locate the **change_scene_button.gd** script. Drag it onto the **Button** node to attach the script.



39 In the **Inspector** of the **Button** node, set the **Level Path** to reference **title_screen.tscn**.

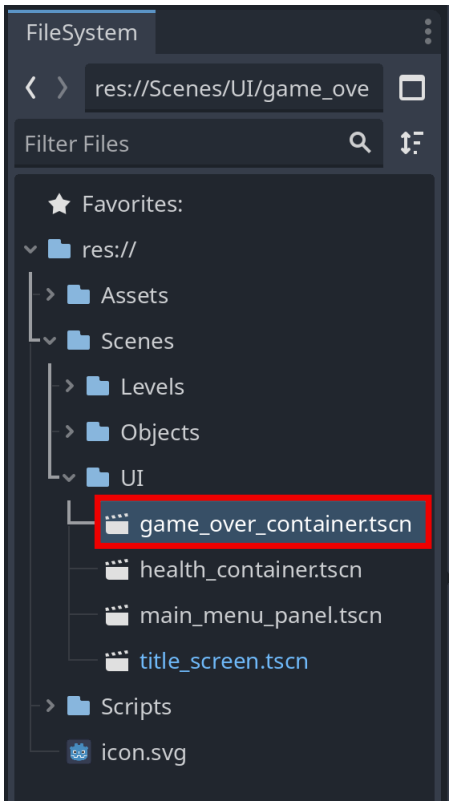


40 Playtest the game. Does the Main Menu button work? Can the user return to the main menu with the game over screen?

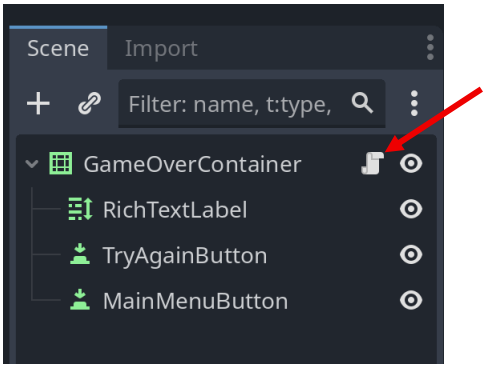


41 The Main Menu button in the Game Over screen is not connected to the title screen.

In **FileSystem**, navigate to **Scenes > UI** and open the **game_over_container.tscn** scene.



42 In **game_over_container.tscn**, open the **script** attached to the **GameOverContainer** node.



43 Under **TODO 4** in **game_over_container.gd,** use the `@export_file()` keyword to declare a `title_screen` variable of type `String`. Pass `"*.tscn"` as the argument to `@export_file()`.

Refer back to **Step 23** as needed.

```
1   extends NinePatchRect
2
3   # ---------------------
4   # TODO 4
5   # Create export_file variable named title_screen
6   # ---------------------
7
8
```

44 Under **TODO 5**, create the `_load_title_screen()` function, which takes no parameters and returns `void`.

Inside the `_load_title_screen()` function, chain together calls to methods `get_tree()` and `call_deferred()`. Pass `"change_scene_to_file"` and `title_screen` as the two arguments.

When `_load_title_screen()` is called, the scene will change to the scene specified by `title_screen`.

Refer back to **Steps 24 – 25** as needed.

```
20  # ---------------------
21  # TODO 5
22  # Create _load_title_screen() function and defer call to change scene
23  # ---------------------
24  # _load_title_screen function
25      # get_tree().........
26
27
```

45

Inside the `_ready()` method under **TODO 6**, use `$` to reference the `MainMenuButton` node and chain together a reference to `pressed` and the `.connect()` method with the argument `_load_title_screen`.

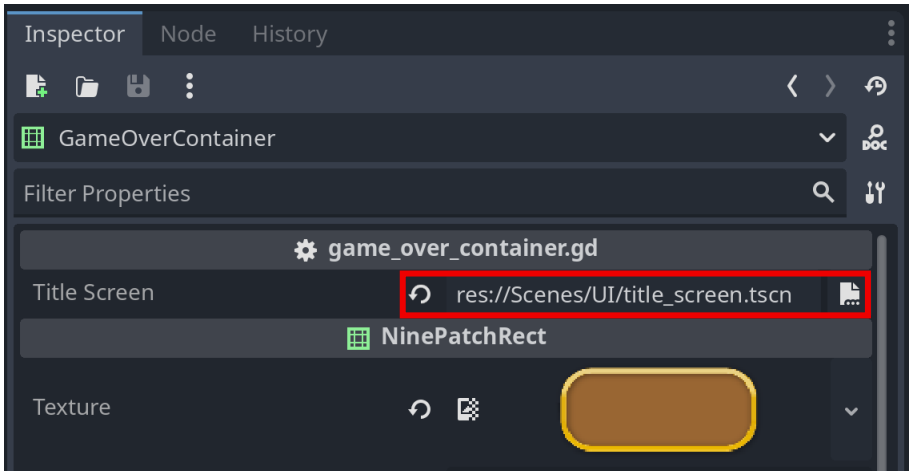
This tells the script to call the `_load_title_screen()` function when the MainMenuButton button is pressed.

Refer to **Step 27** as needed.

```
 9
10 func _ready() -> void:
11     visible = false
12     $TryAgainButton.pressed.connect(LevelManager.restart_level)
13     # ---------------------
14     # TODO 6
15     # Connect $MainMenuButton to _load_title_screen when pressed
16     # ---------------------
17
18     LevelManager.gameOverCanvas = self
19
```

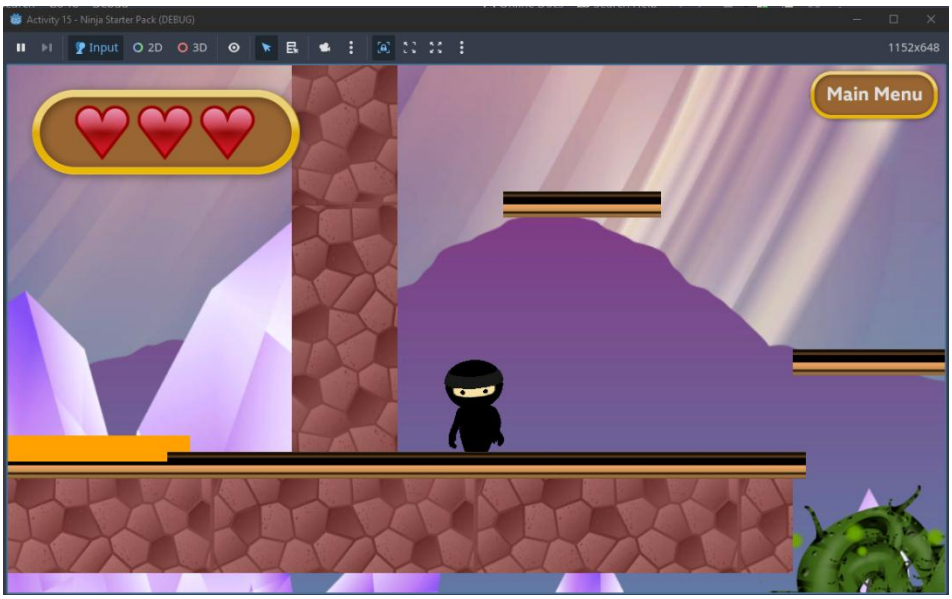
46

In the **Inspector** of the **GameOverContainer** node, set the **Title Screen** to reference **title_screen.tscn**



47 Save the scene and playtest the game.

Both of the Main Menu buttons should now work, loading the title screen UI when clicked on.



48 Check the code in the game_over_container.gd script and update as needed.

```gdscript
extends NinePatchRect

# ---------------------
# TODO 4
# Create export_file variable named title_screen
# ---------------------
@export_file("*.tscn") var title_screen: String


func _ready() -> void:
    visible = false
    $TryAgainButton.pressed.connect(LevelManager.restart_level)
    # ---------------------
    # TODO 6
    # Connect $MainMenuButton to _load_title_screen when pressed
    # ---------------------
    $MainMenuButton.pressed.connect(_load_title_screen)
    LevelManager.gameOverCanvas = self

# ---------------------
# TODO 5
# Create _load_title_screen() function and defer call to change scene
# ---------------------
func _load_title_screen() -> void:
    get_tree().call_deferred("change_scene_to_file", title_screen)
```

Pause for **Sensei Stop #3**!

Congratulations on creating your game with a title screen in Godot! Great job!

Before submitting, check in with a Code Sensei to make sure the health bar UI and its scripts work, then reflect on the following:

- What did you learn about scripting level changes? Creating UI?
- What did you enjoy most when creating this project?
- What was something you found difficult and why?

Reminder: Save your work!