Tutorial One: Creating a Menu

Game Programming Project

By Mariana Neiva Santos Silva

Steps

What you'll learn

In this tutorial you will learn how to set up a basic main menu in unity with buttons to play, quit and options.

Let’s beginning!

Programs used

UNITY

Game Engine

VISUAL STUDIO

Code Editor

ADOBE STOCK

Stock images

What you should already know:

1 A basic understanding of **Unity;**

2 Basic understanding of **C#**

timeline

SETTING UP THE PROJECT

CREATING THE UI

CODING THE SCRIPT

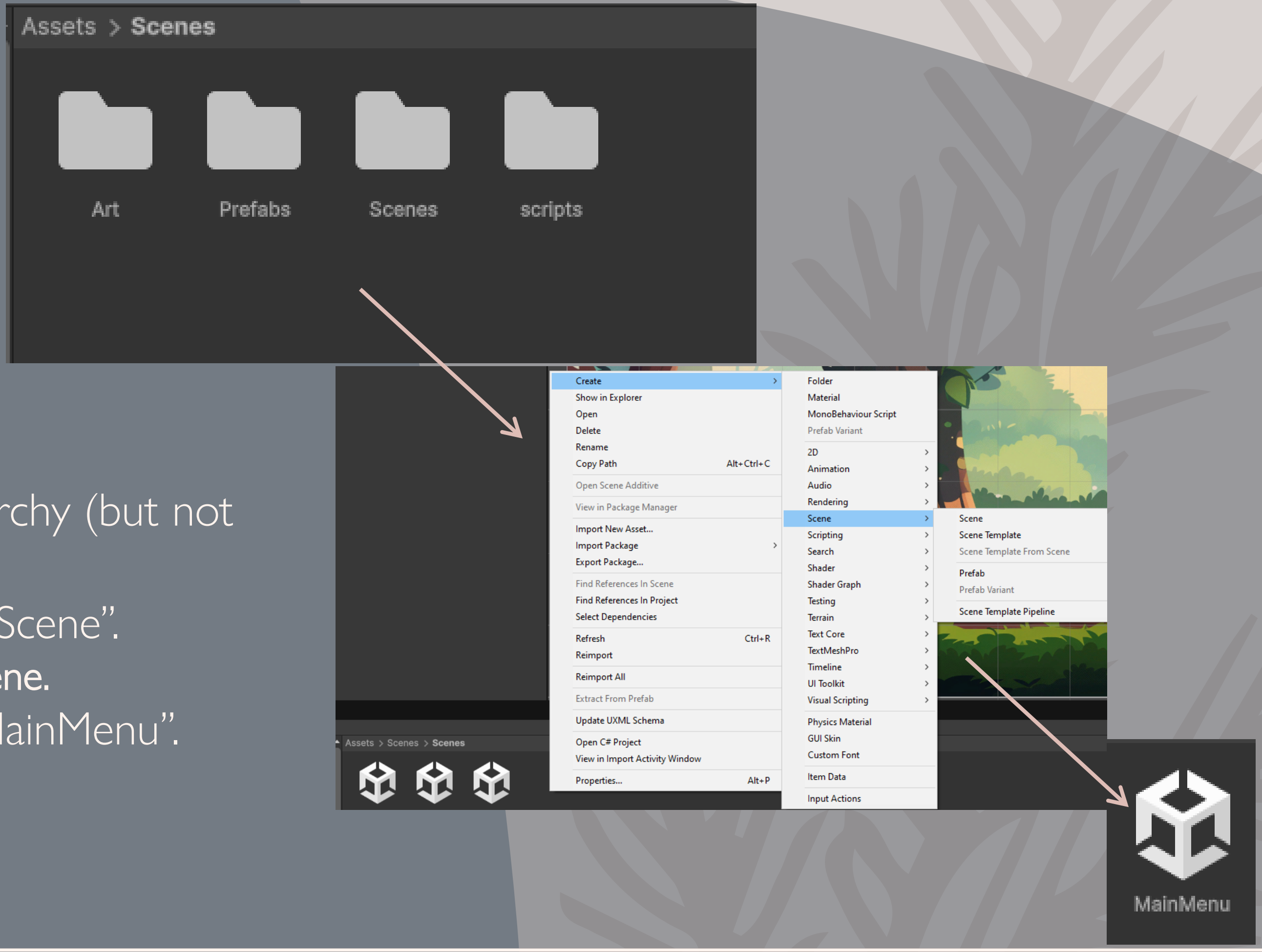
CONNECTING THE BUTTONS TO THE SCRIPT

TESTING

Step 1: Setting up the project.

THE FOLDERS YOU WILL NEED:

- Art: If you want to use sprites and images;
- Prefabs: this will be used in a later tutorial;
- Scenes: Where all the scenes will be;
- Scripts: Where you will keep all your scripts.



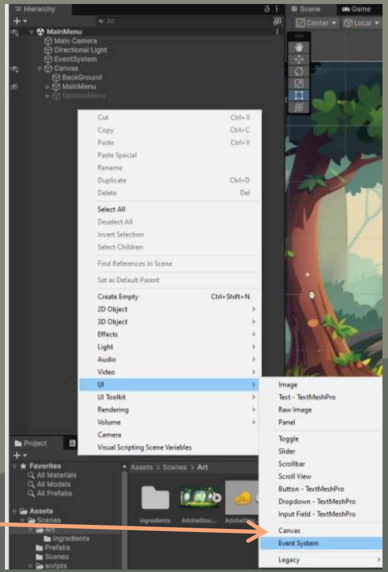
SETTING UP THE MAIN MENU SCENE:

- In the scene folder, right click on Project or the Hierarchy (but not on the scene!).
- Then go to “create” and go to “Scene” then click on “Scene”.
  - Project/Hierarchy > Right Click > Create > Scene > Scene.
- Name it whatever you want. For this I will name it “MainMenu”.

Step 2:Creating a UI Canvas.

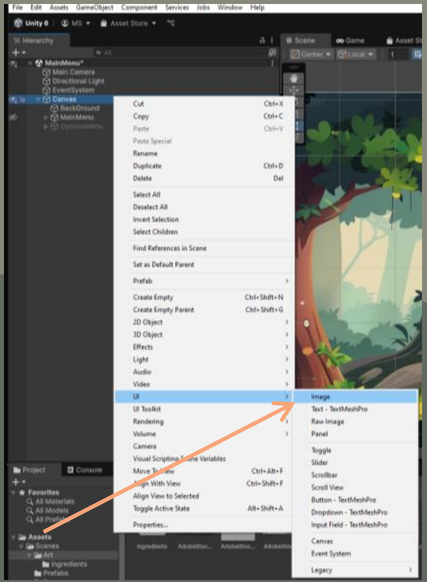
CREATING A CANVAS:

- On the Hierarchy right click and go to UI.
- Then click on canvas.
- Everything we do will go in the Canvas
  - **Hierarchy > Right Click > UI > Canvas**



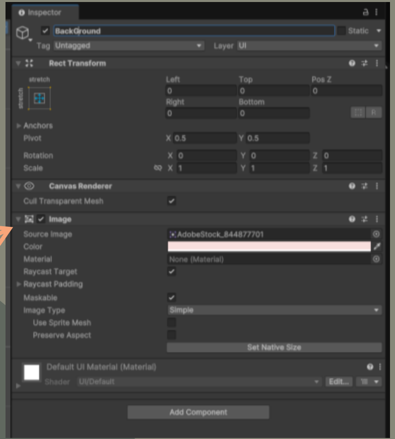
CREATING A BACKGROUND:

- Right Click on the Canvas on the Hierarchy, go to UI;
- The click on Image
  - Canvas > UI > Image



CREATING A BACKGROUND:

- Go to your art folder.
- Select your Background in the Hierarchy.
- In the Inspector, go to image and drag your Image from the folder to "source image".
  - Inspector > Image > Source Image


Inspector
BackGround
Tag: Untagged — Layer: UI
Rect Transform
Left 0, Top 0, Pos Z 0, Right 0, Bottom 0
Pivot X 0.5, Y 0.5
Rotation X 0, Y 0, Z 0
Scale X 1, Y 1, Z 1
Canvas Renderer
Cull Transparent Mesh ✓
Image
Source Image: AdobeStock_844877701
Material: None (Material)
Raycast Target ✓
Maskable ✓
Image Type: Simple
Default UI Material (Material)
Shader: UI/Default

IMPORTANT

Don’t forget to Name things!
I named my canvas and my background, so I can easily find things.
(E.g. StartMenu_Canvas, Background”)

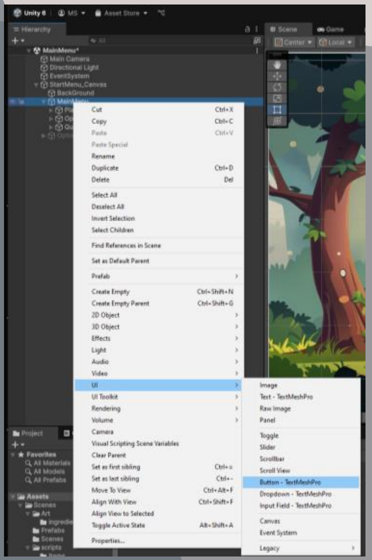
Step 3: Creating the Buttons.

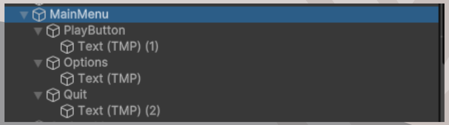
MAIN MENU BUTTONS:

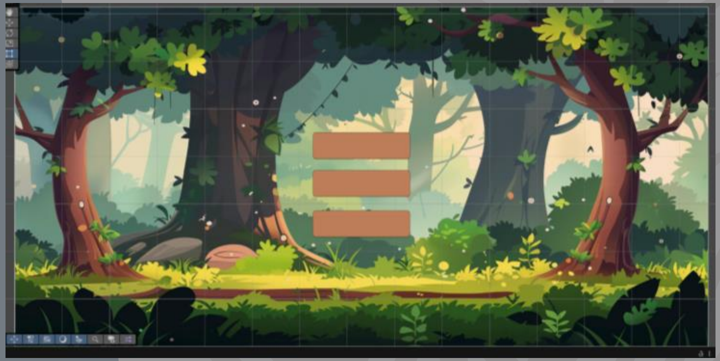
- We will be creating a Start, Options and Quit button.
- First, we will create an Empty game Object to keep everything related to the Main Menu there.
- Create this by right clicking on the hierarchy and clicking on Create Empty.

MAKING THE BUTTONS:

- Inside the empty object – I named it MainMenu – let's create our 3 buttons;
- Name them Play, Options and Quit.
- Inside every button you will have a Text(TMP) this will allow you to display text
- In the Scene Organise the buttons however you would like. Always be careful so that you also move the button and text together.







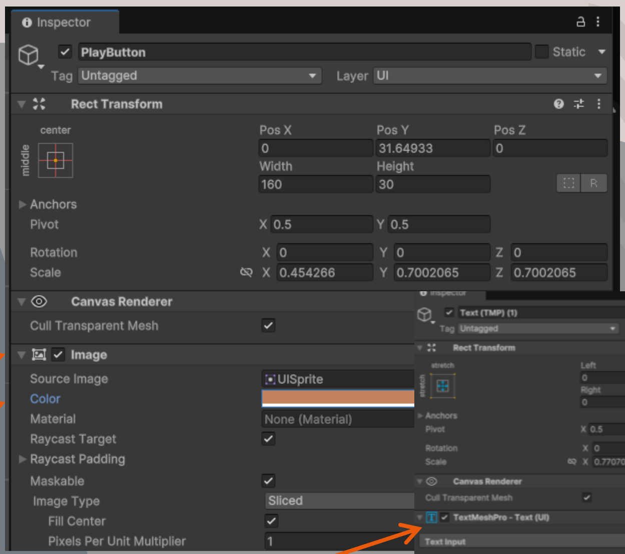
Step 3: Creating the Buttons

ADDING COLOUR:

- Select the button you want to change the colour of and go to the Inspector;
- In image go to colour and pick your desired colour.

ADDING TEXT:

- In the Hierarchy go to the button you want to add text to and go to the Text (TMP).
- In the Inspector go to TextMeshPro – Text(UI);
- In the Text box add whatever text you want to display;
- To change colour, go to Vertex Colour;
- To Change font, go to Font Asset.



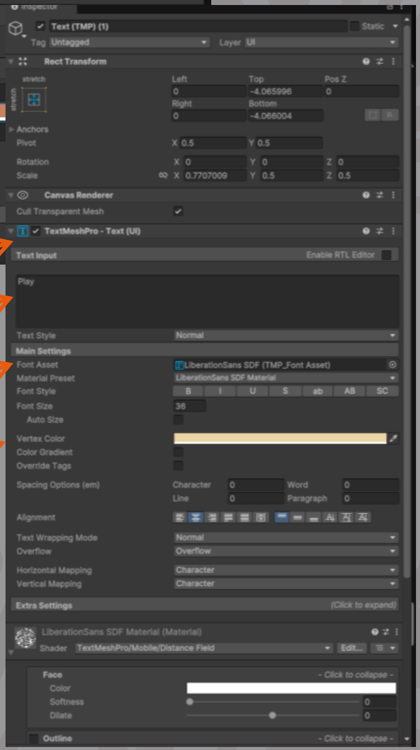
Inspector

PlayButton — Static
Tag Untagged — Layer UI

Rect Transform
center / middle
Pos X 0 — Pos Y 31.64933 — Pos Z 0
Width 160 — Height 30
Anchors
Pivot X 0.5 Y 0.5
Rotation X 0 Y 0 Z 0
Scale X 0.454266 Y 0.7002065 Z 0.7002065

Canvas Renderer
Cull Transparent Mesh ✓

Image ✓
Source Image UISprite
Color
Material None (Material)
Raycast Target ✓
Raycast Padding
Maskable ✓
Image Type Sliced
Fill Center ✓
Pixels Per Unit Multiplier 1



Inspector

Text (TMP) (1) — Static
Tag Untagged — Layer UI

Rect Transform
stretch / stretch
Left 0 — Top -4.065996 — Pos Z 0
Right 0 — Bottom -4.066004
Anchors
Pivot X 0.5 Y 0.5
Rotation X 0 Y 0 Z 0
Scale X 0.7707009 Y 0.5 Z 0.5

Canvas Renderer
Cull Transparent Mesh ✓

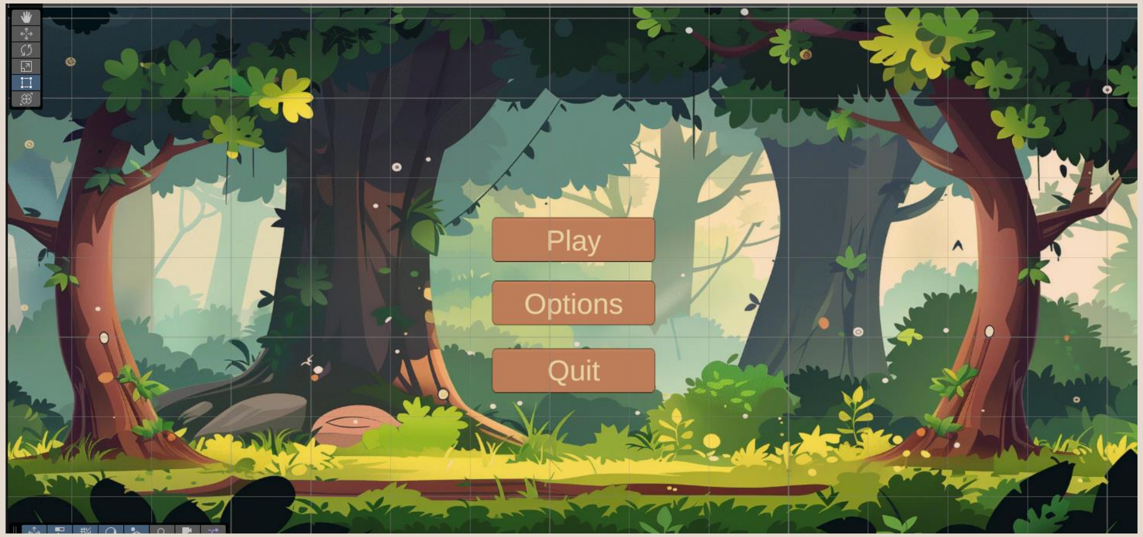
TextMeshPro - Text (UI) ✓
Text Input — Enable RTL Editor
Play
Text Style Normal
Main Settings
Font Asset LiberationSans SDF (TMP_Font Asset)
Material Preset LiberationSans SDF Material
Font Style B I U S ab AB SC
Font Size 36
Auto Size
Vertex Color
Color Gradient
Override Tags
Spacing Options (em) Character 0 Word 0 Line 0 Paragraph 0
Alignment
Text Wrapping Mode Normal
Overflow Overflow
Horizontal Mapping Character
Vertical Mapping Character
Extra Settings (Click to expand)

LiberationSans SDF Material (Material)
Shader TextMeshPro/Mobile/Distance Field — Edit...
Face - Click to collapse -
Color
Softness 0
Dilate 0
Outline - Click to collapse -

IMPORTANT

If all your buttons look the same make one and make copies, then you only need to change the text and the names.

What you should have now



Let’s check in our Progress!

SETTING UP THE PROJECT

CREATING THE UI

CODING THE SCRIPT

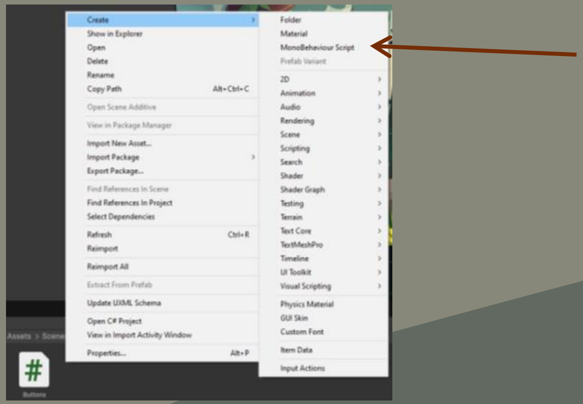
CONNECTING THE BUTTONS TO THE SCRIPT

TESTING

Step 4: The Script

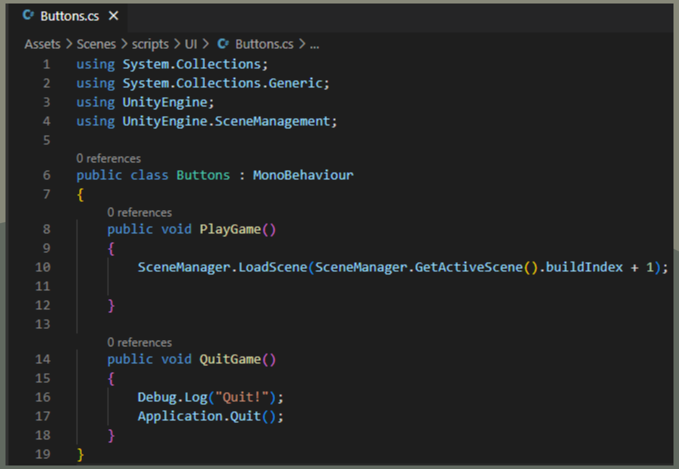
CREATING THE SCRIPT:

- In your Script Folder right click and go to create and click on MonoBehaviour Script;



THE SCRIPT

- The script bellow will control your buttons.
- The options button will use a different technique.


Buttons.cs

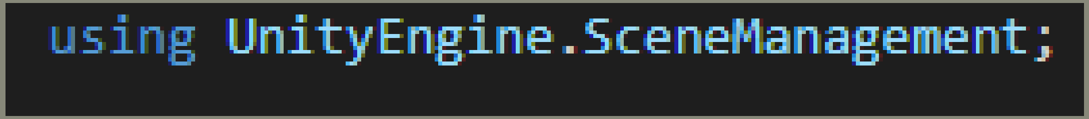
Assets > Scenes > scripts > UI > Buttons.cs > ...

```csharp
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

0 references
public class Buttons : MonoBehaviour
{
    0 references
    public void PlayGame()
    {
        SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);

    }

    0 references
    public void QuitGame()
    {
        Debug.Log("Quit!");
        Application.Quit();
    }
}
```

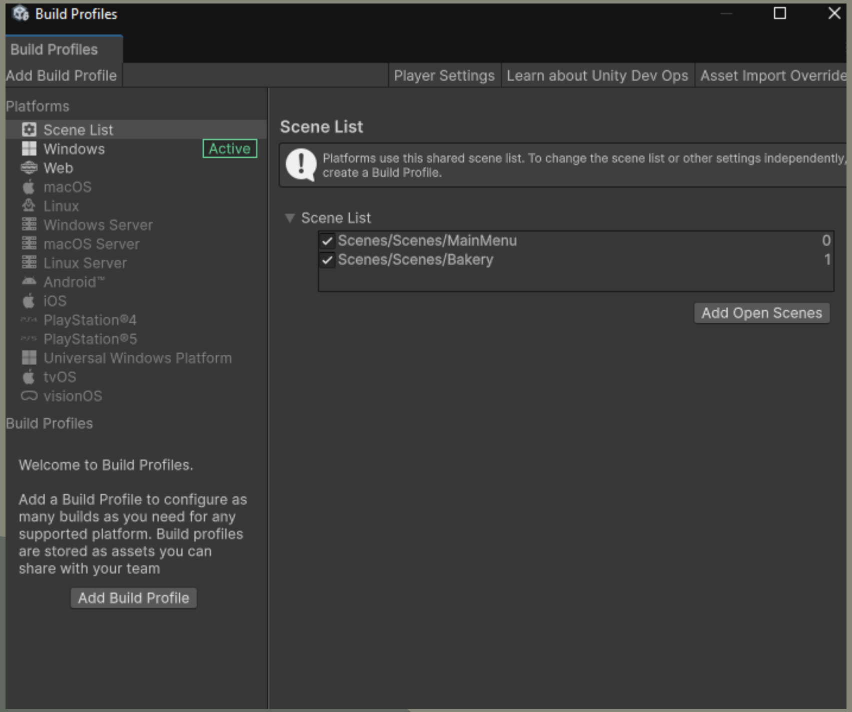
Step 4: Understanding the Script

SCENE MANAGEMENT

- We first, need to tell Unity that in this script we want to manage scenes with the line:
- *"using UnityEngine.SceneManagement;"*
- It will allow us to use the build settings scene list:



```
using UnityEngine.SceneManagement;
```



Scene List

Platforms use this shared scene list. To change the scene list or other settings independently, create a Build Profile.

Scene List

```
✓ Scenes/Scenes/MainMenu    0
✓ Scenes/Scenes/Bakery      1
```

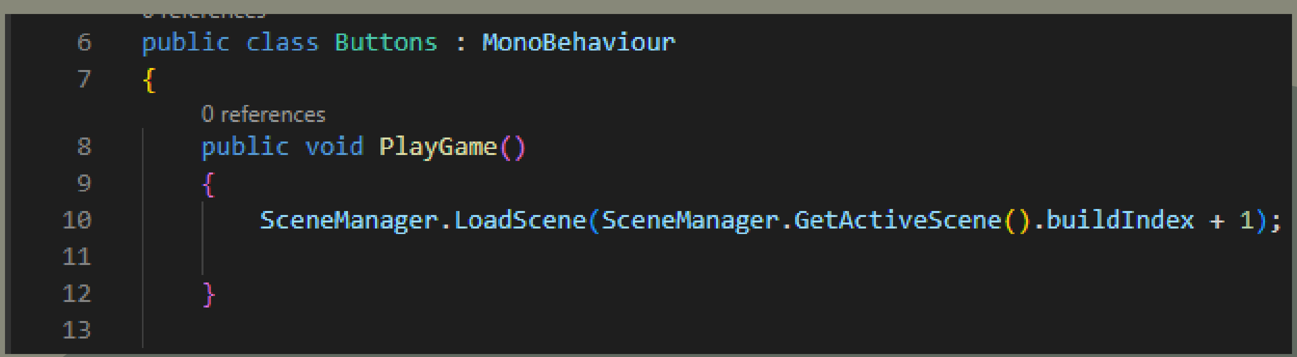
Welcome to Build Profiles.

Add a Build Profile to configure as many builds as you need for any supported platform. Build profiles are stored as assets you can share with your team

Step 4: Understanding the Script

PLAYGAME()

- We now need to make a public void called "PlayGame()" :
- *"public void PlayGame()"*
  - In this we will add this line:
- *SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);*
  - This command loads the next scene in unity.
- SceneManager.GetActiveScene()
  - This gets the scene currently active.
- *buildIndex + 1*
  - This gets the index in Unity's build settings that we previously talked about and will load the next one on it.

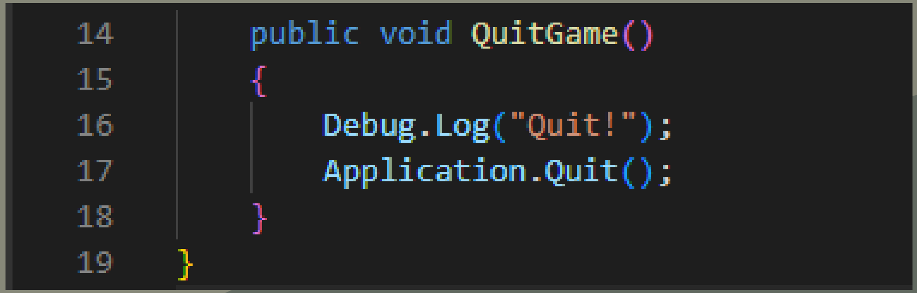


```
0 references
public class Buttons : MonoBehaviour
{
    0 references
    public void PlayGame()
    {
        SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);

    }

```

Step 4: Understanding the Script

QUITGAME()

- We now need to make a public void called "QuitGame()" :
- *"public void QuitGame()"*
  - In this line we will add:
- *" Debug.Log("Quit!");"*
  - This will allows us to see if the button works in the unity Game, it will display Quit in the console.
- " Application.Quit(); "
  - This will make the built version of the game close.



```
    public void QuitGame()
    {
        Debug.Log("Quit!");
        Application.Quit();
    }
}
```

Additional Information:

VOID:

- What is a function?
  - Is code that does a task and then returns to a value.
- Usually, functions return a value:
  - An Integer: whole number
  - A Boolean: true or false
- A void is a Function that has no return value.

CLASS:

- What is a class?
  - They are used to define properties and behaviours.
  - I see them as categories. Some people see them ass blueprints.

PUBLIC VS PRIVATE METHODS

- What is a Method?
  - It's a chunk of code that has statements.
- Private methods can only be accessed in its class, and it is hidden from the rest of the code, you cannot see/change them in Unity.
- Public methods mean that it can be found by other classes and scripts, you can also see/ change it it in Unity itself.

Let’s check in our Progress!

SETTING UP THE PROJECT

CREATING THE UI

CODING THE SCRIPT

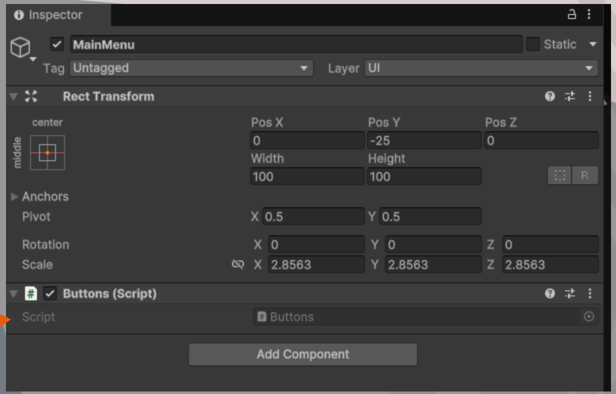
CONNECTING THE BUTTONS TO THE SCRIPT

TESTING

Step 5: Connecting the buttons.

ADDING THE SCRIPT

- We're going to add the script we just went through to the Empty Game object we created called MainMenu.



Inspector

MainMenu — Static

Tag: Untagged — Layer: UI

Rect Transform

center / middle

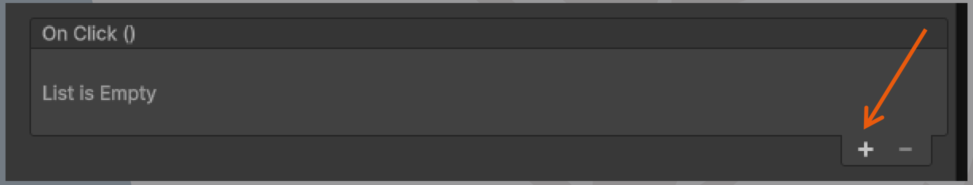
	Pos X: 0	Pos Y: -25	Pos Z: 0
	Width: 100	Height: 100	
Anchors			
Pivot	X 0.5	Y 0.5	
Rotation	X 0	Y 0	Z 0
Scale	X 2.8563	Y 2.8563	Z 2.8563

Buttons (Script)

Script: Buttons

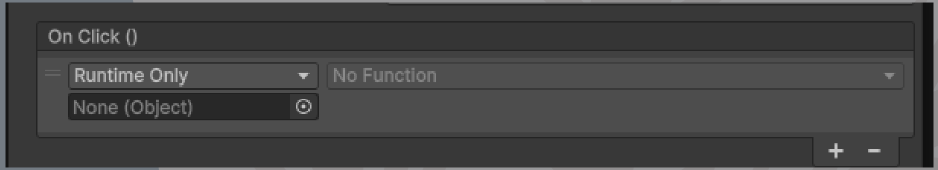
ADDING ON CLICK

- In the Button you are editing go to the inspector and press the + button in the On Click() option.



On Click ()

List is Empty



On Click ()

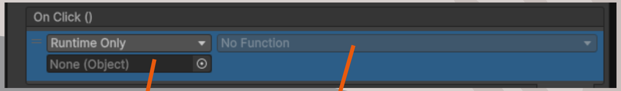
Runtime Only — No Function

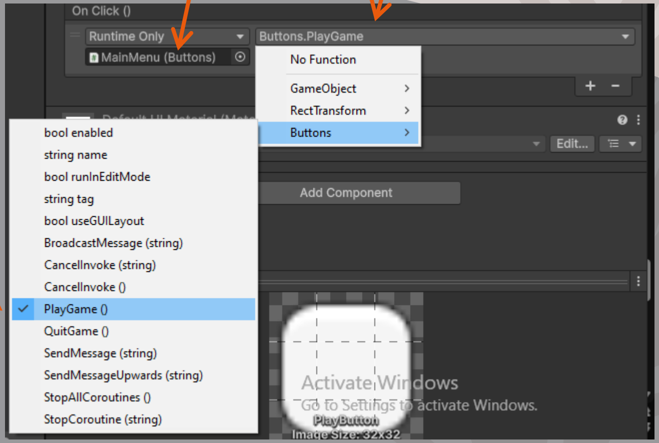
None (Object)

Step 5: Connecting the buttons.

PLAY BUTTON

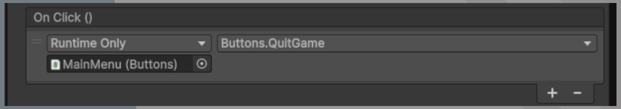
- Go to the play button in the Inspector go to OnClick().
- We are going to drag the MainMenu Object from the previous slide to the OnClick().
- Then we option that says “No function” and go to “Buttons” (this is a reference to our script) and then pick the function PlayGame().
- This is why we created a Public Void and not a Private Void. Since it is Public we can use it in Unity.

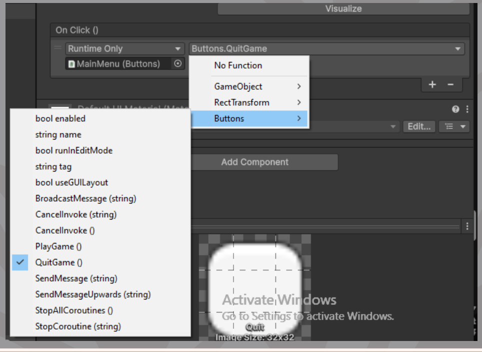




QUIT BUTTON

- We now will do the same thing for the Quit button but instead we will pick the function QuitGame().





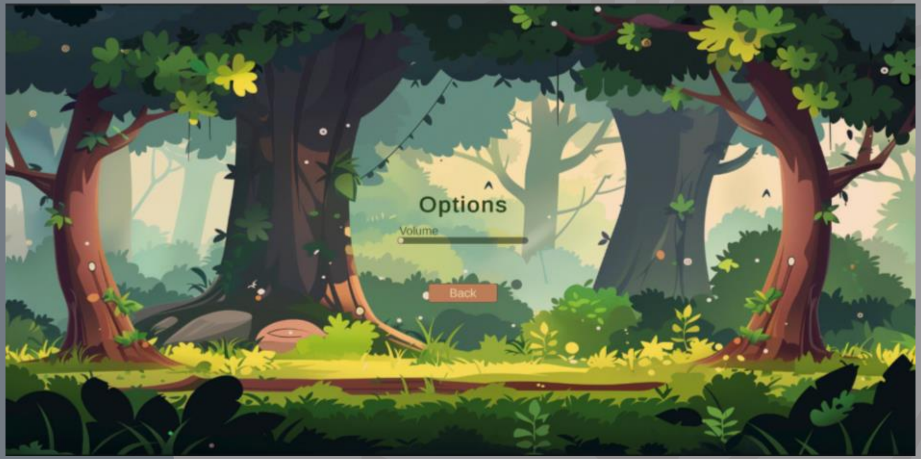
Step 5: Additional - Options button.

HOW IT WORKS:

- The options button will work differently. It isn't a line in our code.
- It simply makes the Empty game object "MainMenu" be invisible and the Empty Game object "OptionsMenu" be visible.

SETTING UP

- In your canvas create an empty game object called "OptionsMenu".
- Add whatever buttons you want to it. It might be easier to make turn the visibility of the "MainMenu" off to do this.



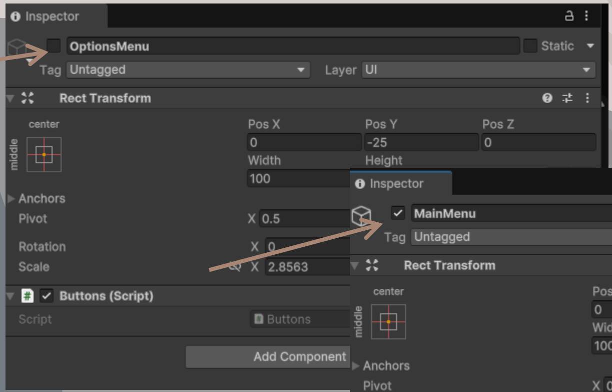
Step 5: Additional - Options button.

SETTING UP:

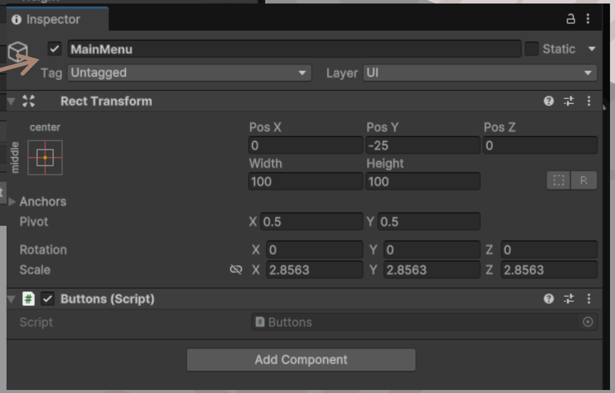
- In the inspector disable the "OptionsMenu"
- Make sure that at this point your main menu is visible and enabled!

OPTIONS BUTTON

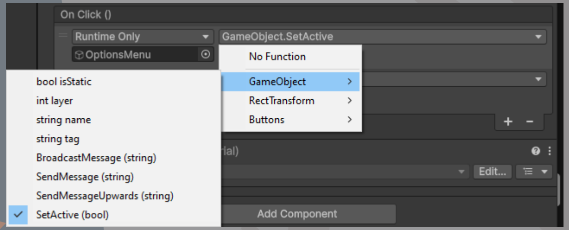
- Add two On click() to the options button.
- Drag the "OptionsMenu" to the first one and the "Mainmenu" to the second one.
- Set them both to Game Object SetActive bool. This will make it so you can make the Enabling or Disabling of the objects true or false.
- Now set the first button to Enable so that the Options menu shows up when the player clicks it
- On the second one disable the "MainMenu" so it disappears.



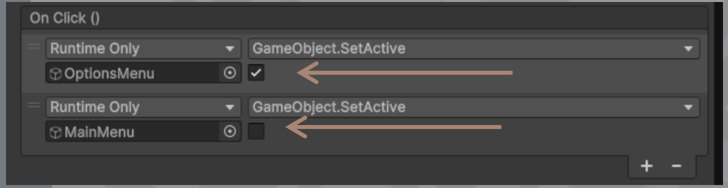
Inspector
OptionsMenu — Static
Tag: Untagged — Layer: UI
Rect Transform
center / middle
Pos X: 0 — Pos Y: -25 — Pos Z: 0
Width: 100 — Height
Anchors
Pivot: X 0.5
Rotation: X 0
Scale: X 2.8563
Buttons (Script) — Script: Buttons
Add Component



Inspector
✓ MainMenu — Static
Tag: Untagged — Layer: UI
Rect Transform
center / middle
Pos X: 0 — Pos Y: -25 — Pos Z: 0
Width: 100 — Height: 100
Anchors
Pivot: X 0.5 — Y 0.5
Rotation: X 0 — Y 0 — Z 0
Scale: X 2.8563 — Y 2.8563 — Z 2.8563
✓ Buttons (Script) — Script: Buttons
Add Component



On Click ()
Runtime Only — GameObject.SetActive
OptionsMenu



On Click ()
Runtime Only — GameObject.SetActive
OptionsMenu — ✓
Runtime Only — GameObject.SetActive
MainMenu — ☐

IMPORTANT

You can use the same logic to make a back button!

Let’s check in our Progress!

SETTING UP THE PROJECT

CREATING THE UI

CODING THE SCRIPT

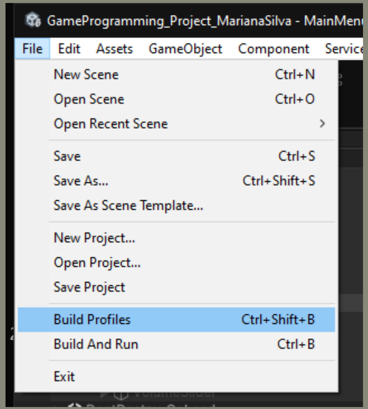
CONNECTING THE BUTTONS TO THE SCRIPT

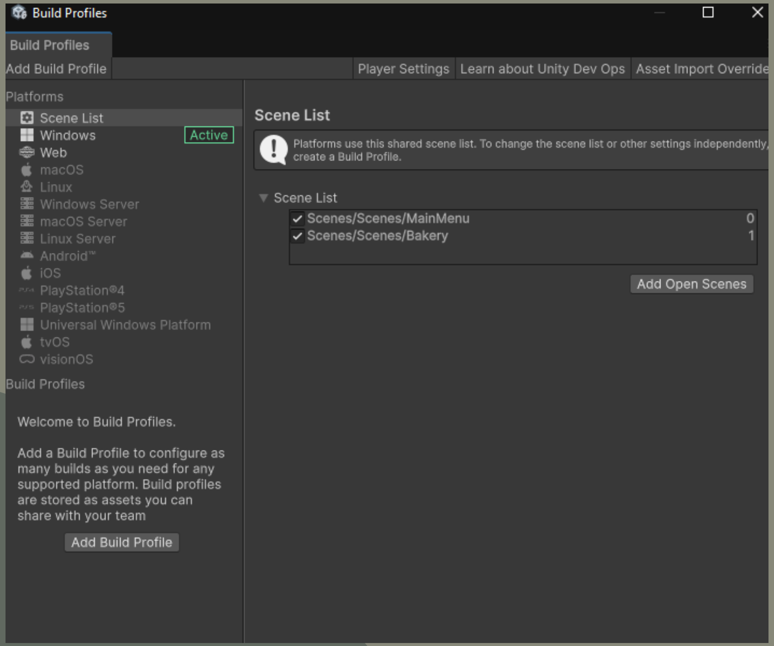
TESTING

Step 6: Testing.

BEFORE WE TEST!

- Let's add our scenes to the build profile scene list.
- You can find this in File > Build Profiles > Scene List.
- The main menu should be your scene zero!
- Then add whatever scene by dragging them from the project
- Whatever scene you add after the main menu will play when the player hits Play.





Scene List

Platforms use this shared scene list. To change the scene list or other settings independently, create a Build Profile.

▾ Scene List

✓ Scenes/Scenes/MainMenu	0
✓ Scenes/Scenes/Bakery	1

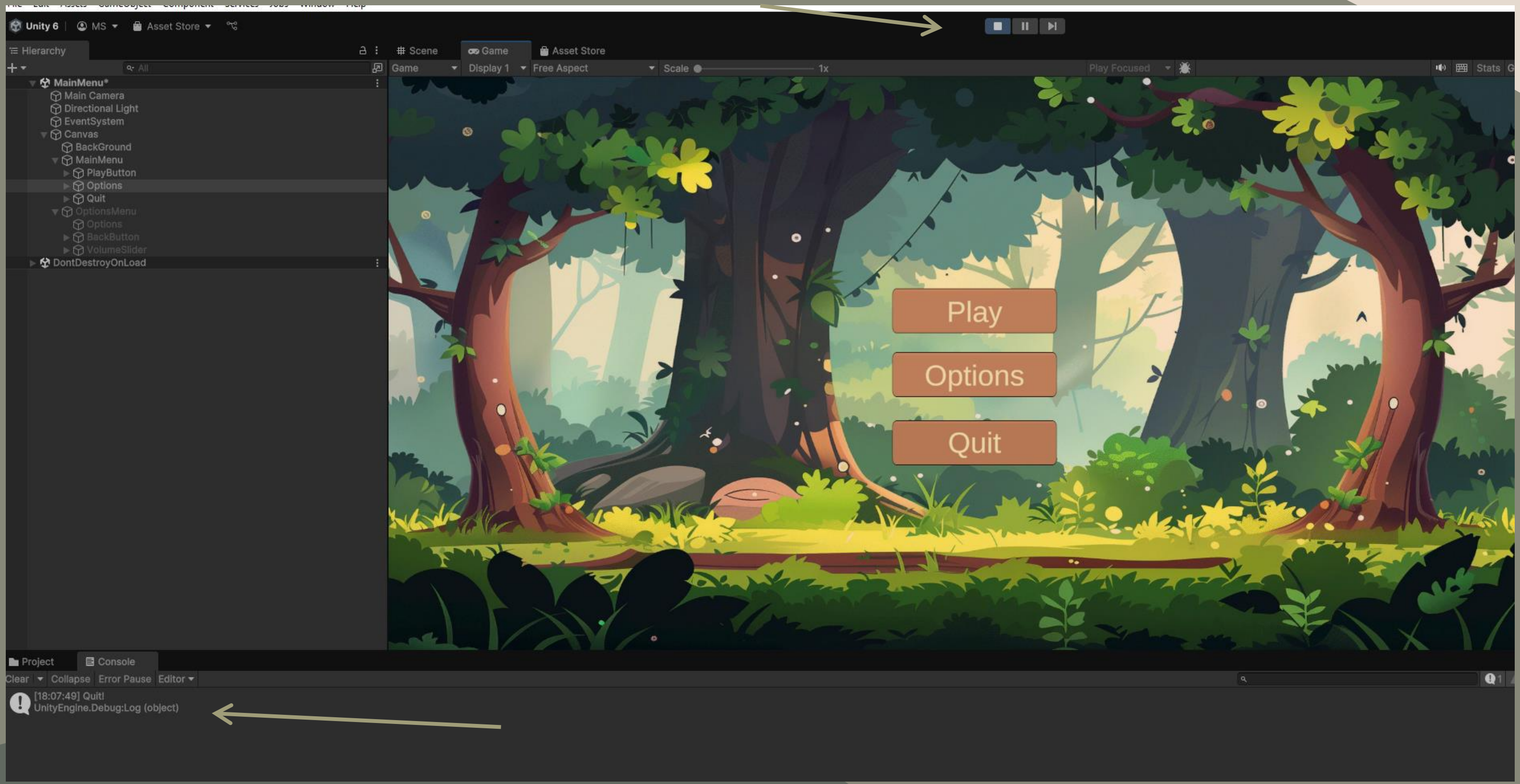
Welcome to Build Profiles.

Add a Build Profile to configure as many builds as you need for any supported platform. Build profiles are stored as assets you can share with your team

Step 6: Testing.

PLAYING OUR GAME:

- Now let's test our work!
- All the buttons should be working.
- Remember that the quit button will show a Debug message in the console of "Quit!"



```
[18:07:49] Quit!
UnityEngine.Debug:Log (object)
```

Let's check in our Progress!

SETTING UP THE PROJECT

CREATING THE UI

CODING THE SCRIPT

CONNECTING THE BUTTONS TO THE SCRIPT

TESTING

Congratulations!

You now can make a basic main menu UI in unity!

Thank you