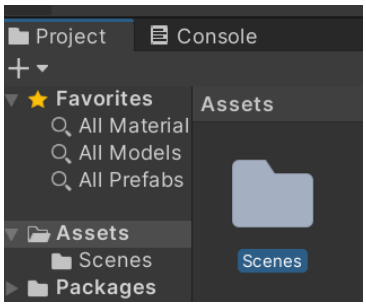
Unity Main menu and level select tutorial

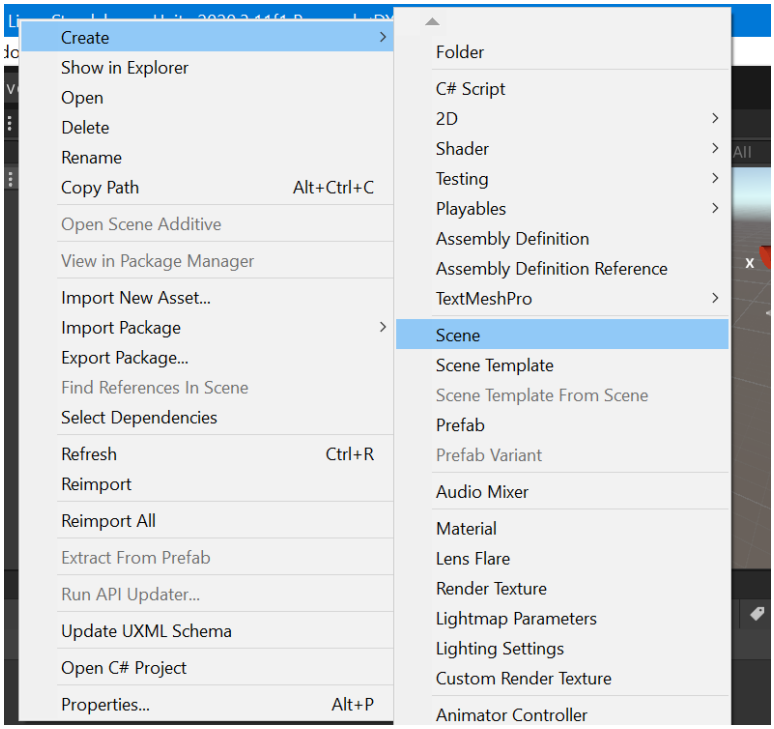
Open a new Unity scene for the menu

To do this

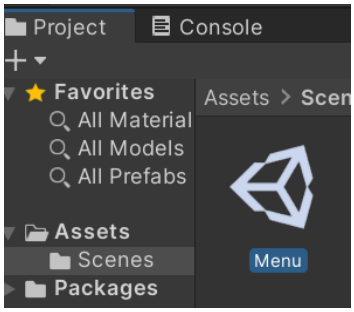
1. Click on the scenes folder to open it



2. Right click in the folder and click on Create > Scene
3. Give the new scene an appropriate name for a menu



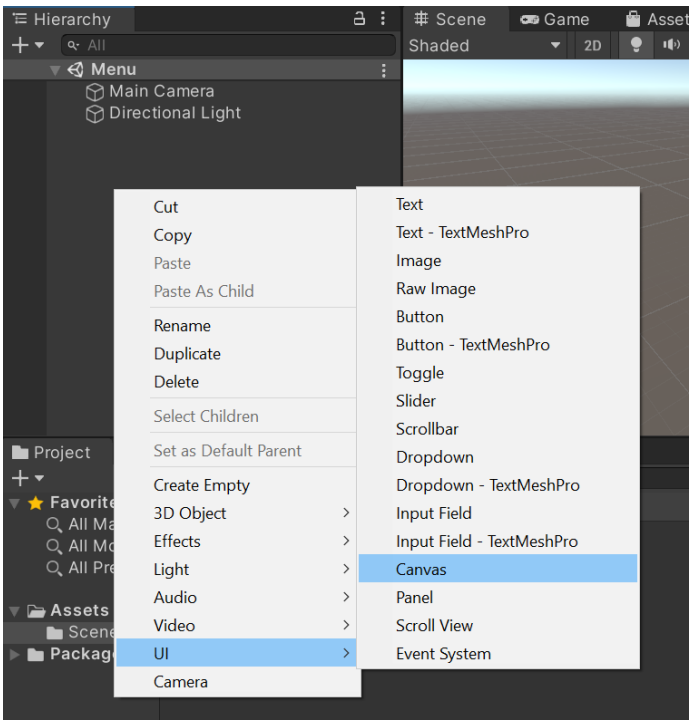
4. Click on the scene to open it



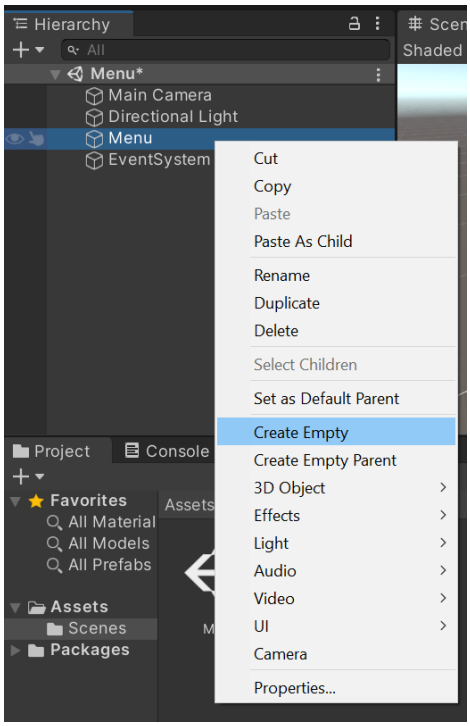
Create a canvas with 2 empty gameobjects attached

To do this

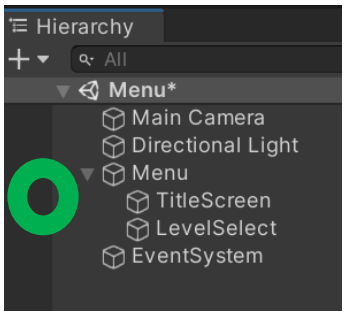
1. Right click in the hirearchy and click UI > Canvas
2. Rename the canvas to Menu



3. Make sure the Canvas is highlighted and then right click in the hirearchy and click “Create Empty”
4. Name this Gameobject TitleScreen



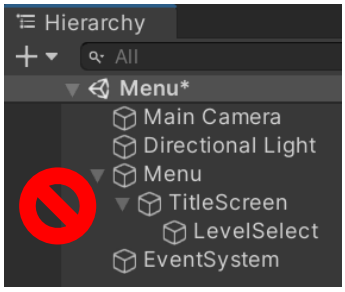
5. Repeat the third step making sure that the canvas is the object selected
6. Name this Gameobject LevelSelect



Make sure that the TitleScreen and LevelSelect are further to the right than everything else and are directly underneath Menu.

If they are not click and drag LevelSelect and TitleScreen into the Menu Canvas.

This is because they will become children objects of the Menu and means that they are connected to the menu.

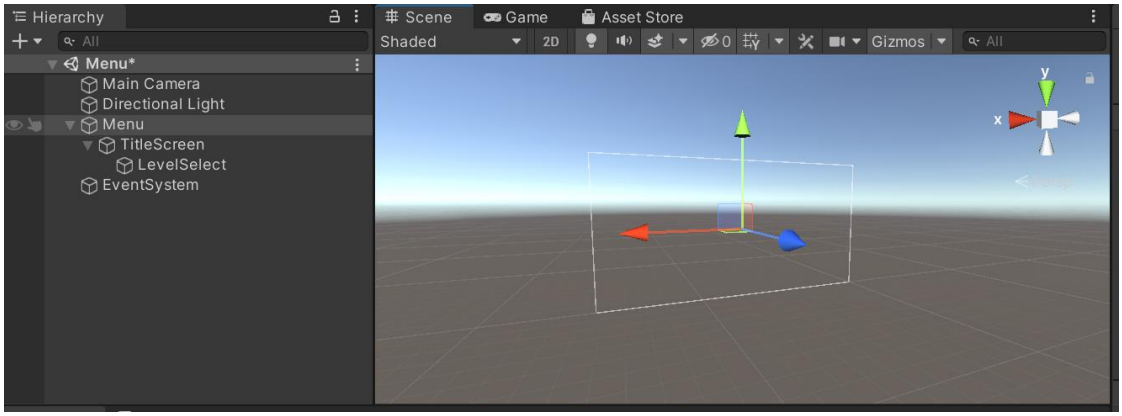


Be sure not to make one of the empty Gameobjects a child of another as this will make them linked which will not allow us to swap between them later on.

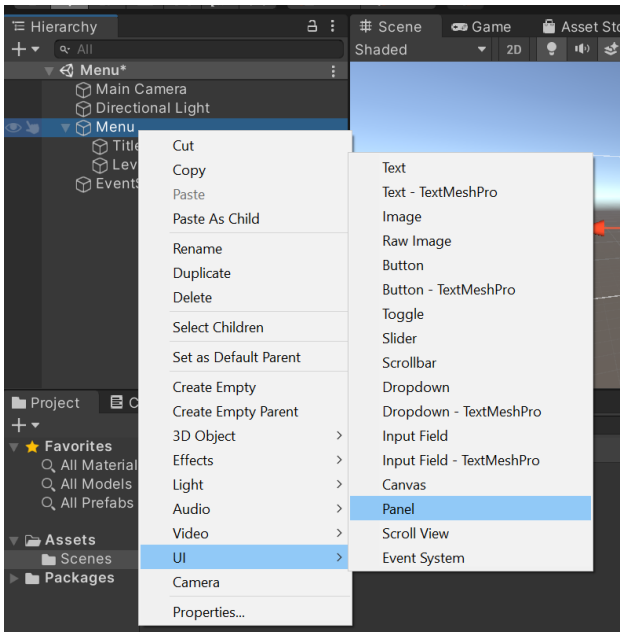
To fix this click and drag the Gameobject that has become a child into the Menu Canvas.

Add a coloured background to our Menu

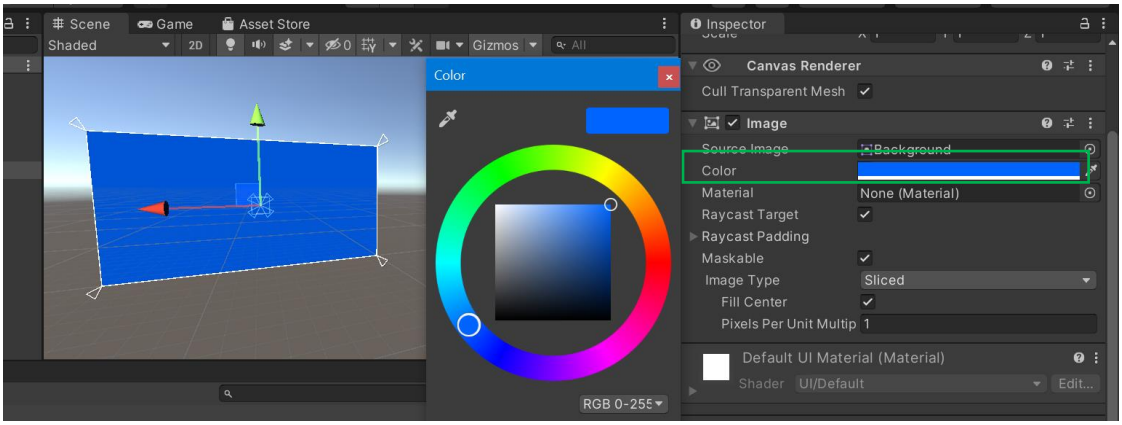
1. In the hierarchy and double click on Menu (this will move the camera so we can see our canvas)



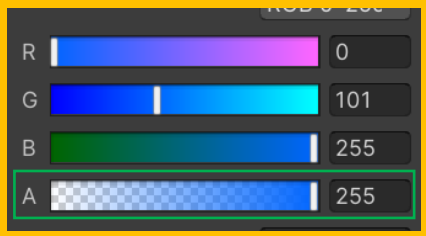
2. Right click on Menu in the hierarchy and click UI > Panel
3. Rename the Panel to Background
4. Drag the Background to be above TitleScreen and LevelSelect in the hierarchy



5. In the Inspector you click the box next to Colour and choose what colour you want the background to be



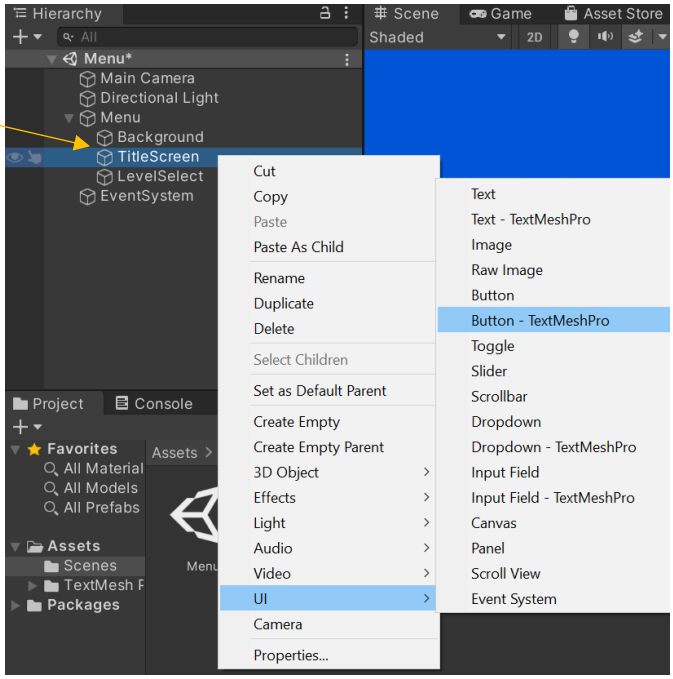
For this tutorial we will make a menu with a static colour background with nothing else going on. As a result, we will want to have our opacity (Marked with an A) to be set to 255 so we cannot see anything else. More advanced menus can make use of a game camera to show of something in the background like an environment or character.



Making the button for our title screen

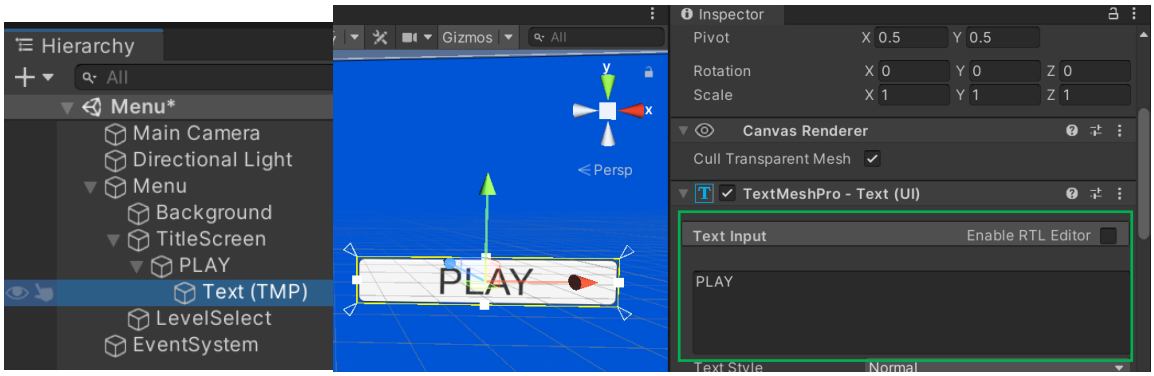
1. While TitleScreen is highlighted right click on UI > Button – TextMeshPro
2. Rename this button to PLAY

If you cannot see your button on your canvas, make sure the background is set to above the TitleScreen and LevelSelect in the hierarchy.

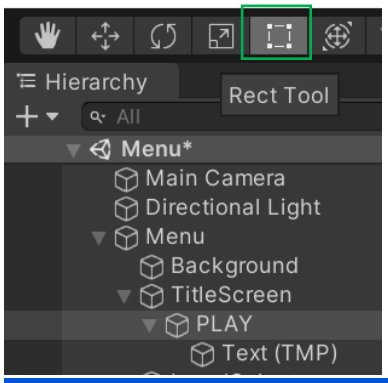


We are using a TextMeshPro (TMP) button as it has more customisation abilities allowing for better looking Menu's however a standard button will also work fine if you just want to make a functioning button.

3. Click the dropdown arrow on the PLAY button and click the Text underneath it
4. In the Inspector scroll down to where it says Text Input and replace Button with PLAY

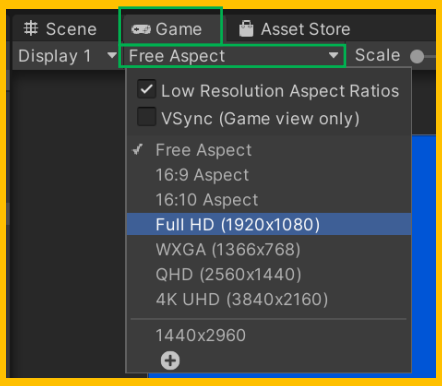


5. Click back on the PLAY in the hierarchy go to the Rect tool (Square with dots) and resize the button using the dots to the size you want it



Remember to have your Game view set to the correct ratio otherwise your UI elements will be resized when you eventually do.

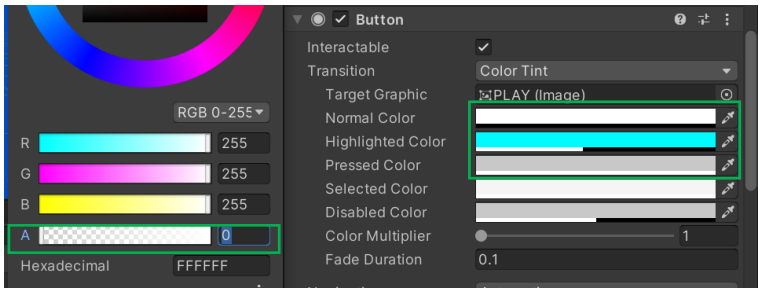
To do this click Game > Free Aspect > select the ratio you want.



Customising the button

This step is optional and won't affect the function of the button therefore I will show various things that you can do to your button and text to make them stand out, feel free to experiment with visual settings until you are happy.

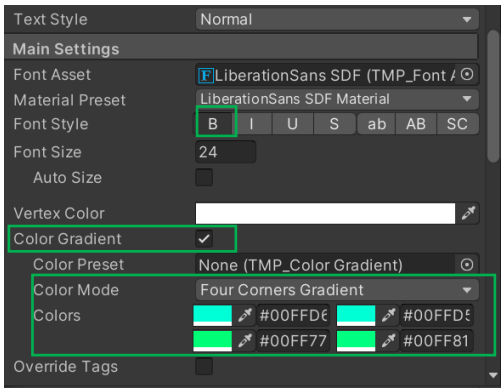
1. Make the button layer clear unless it is about to be clicked



Set the Normal colour on the button to have 0 opacity then change its Highlighted colour to have a small amount of opacity (Around 110) and the pressed colour to an opacity above that.

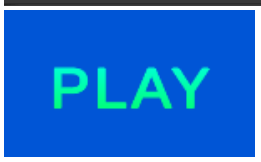
Remember you can change the colour the button will change to and use play mode to preview how it will look getting clicked.

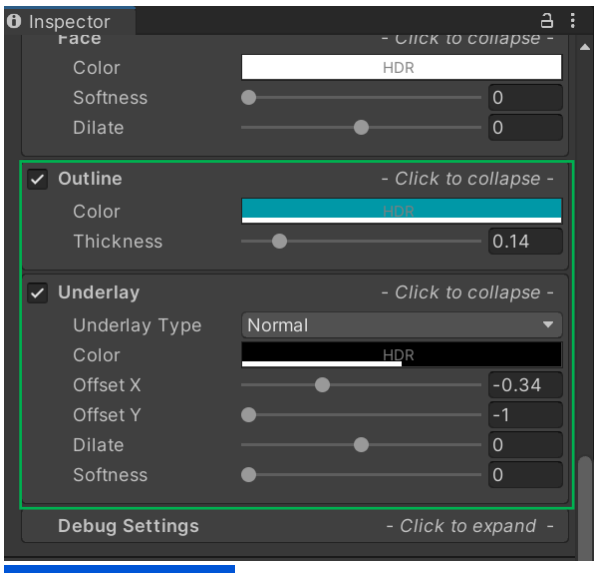
2. Making the text on the button stand out more



Firstly, setting the text to bold can make it stand out more also because we used a TMP button we can add a gradient to our button so it changes colour throughout.

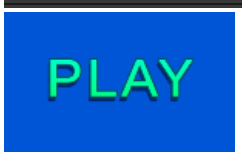
To do this tick the colour gradient box and assign the colours you want the text to have





Another detail you can add to your text is to make it have an outline so that the colours don't blend into the button or background and an underlay to make it pop more.

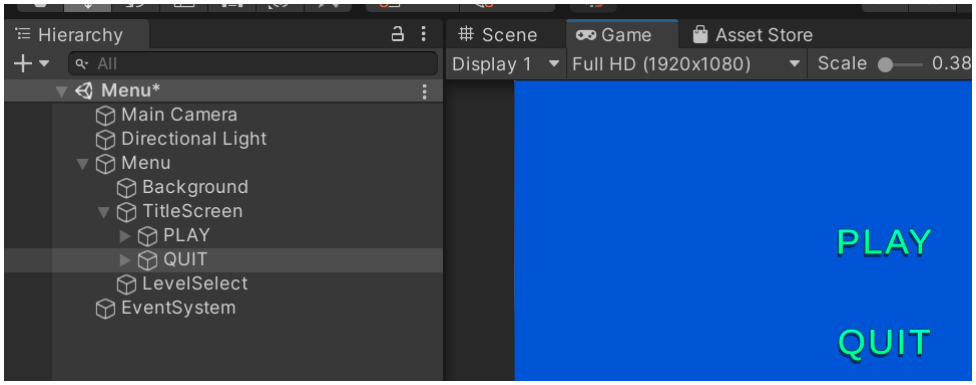
There are no specific settings you need just apply these features if you want and adjust them to your liking.



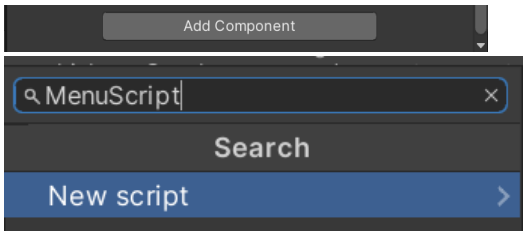
3. Everything altogether

Creating a quit button

1. Duplicate the Play button and rename it to QUIT
2. Adjust the text so the button says QUIT instead of PLAY

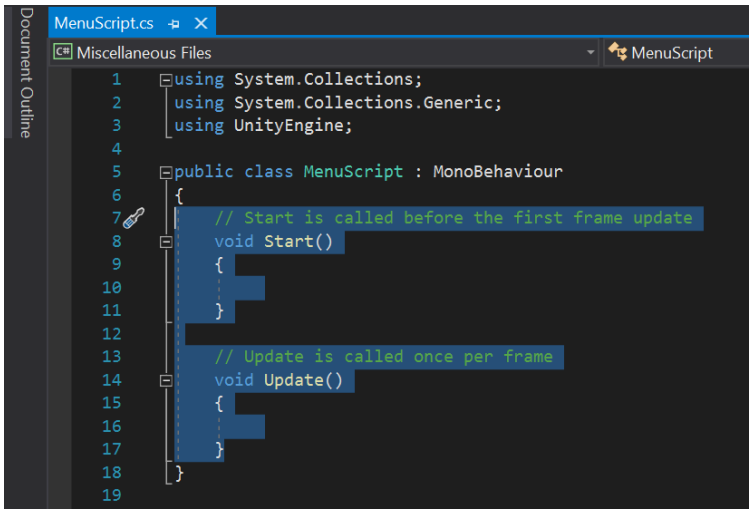


3. Click the QUIT button and go down to Add component in the Inspector
4. Type MenuScript and click New Script then Create and add then open the script



We will be using this script for all our buttons so we will duplicate our buttons later to save some time

5. Once the script is open delete the highlighted section as we won't be needing them for this particular script



6. Copy this code

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

public class MenuScript : MonoBehaviour
{
    public void QuitGame()
    {
        Debug.Log("QUIT");
        Application.Quit();
    }
}
```

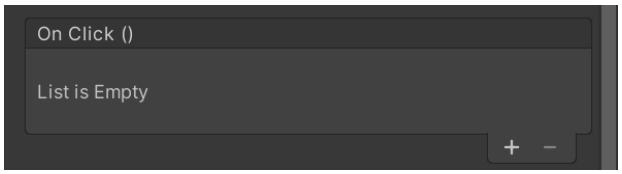
Here we are making a void, this means that when we call upon the function QuitGame anything that is inside that void will happen.

The reason we use void is because we don't need anything back from the code such as numbers and values. We just want it to do what we have told it to do.

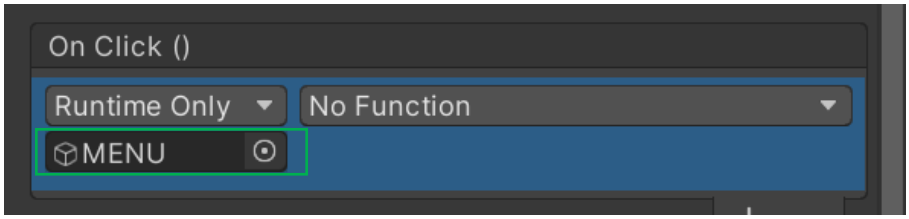
Application.Quit(); means that no matter what the game is doing it will stop playing and will close however while editing our game this will not work so to ensure the code is working properly we use Debug.Log("Quit") as this sends us a message saying quit when the code is triggered.

Debug text is used all the time for testing if code is working and is very common when troubleshooting problems

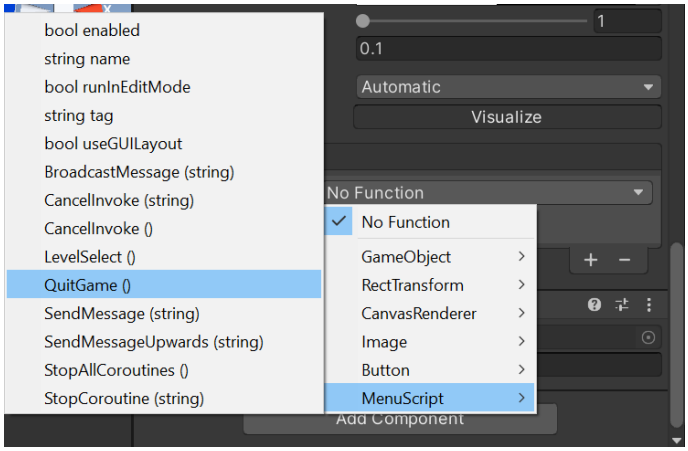
7. Save this code and return to Unity
8. Go to the OnClick() in the Inspector



9. Click the +
10. Click and drag the QUIT in the hierarchy to where it says None (Object) in the Inspector

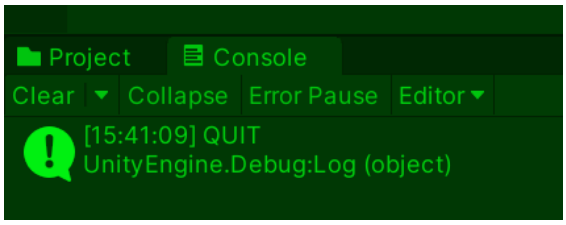


11. Now click No Function > MenuScript > QuitGame()



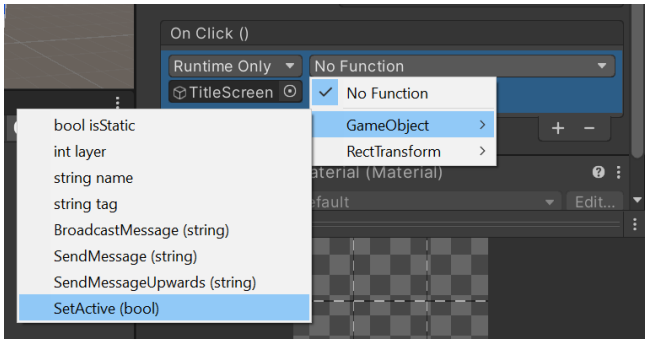
Because we attached the MENU which had the script attached to it both the script and the function we needed are able to be found

12. To test this, go into play mode and click the quit button, you should see a quit message appear in the console

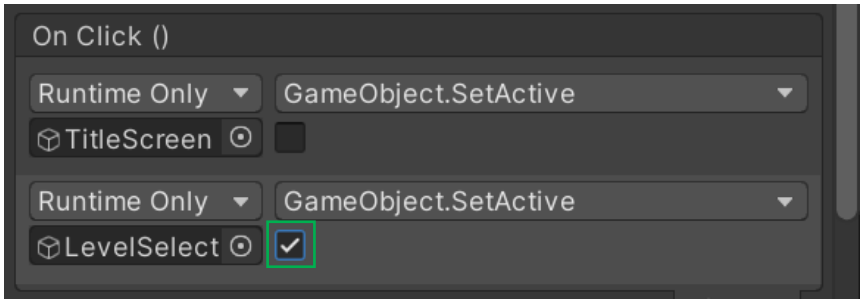


Switching to the level select menu

1. Click the PLAY button and go to the OnClick() in the Inspector as before, click the + and move the TitleScreen into the None (Object)
2. Click No function > GameObject > SetActive (bool)



3. Click the + again and this time drag the LevelSelect into the inspector and repeat step 2
4. Tick the box that has appeared



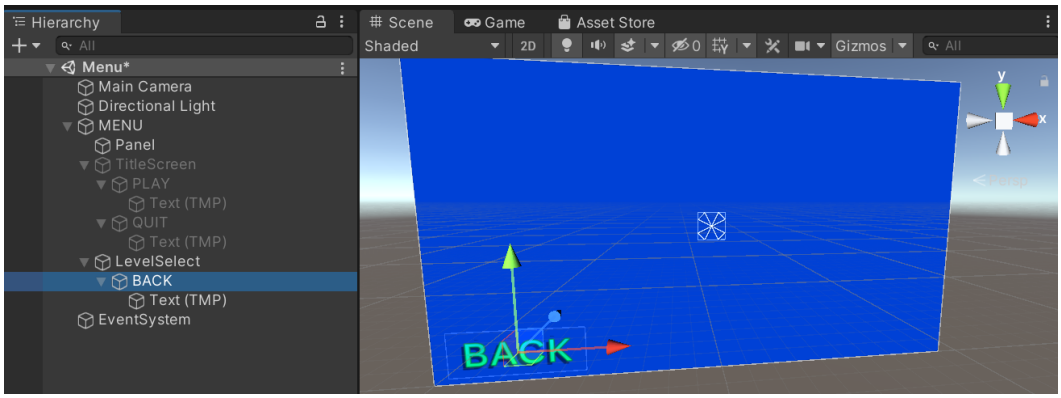
What we are getting Unity to do here is to turn off the TitleScreen and turn on the LevelSelect so we are essentially switching between them.

If you were to test this out in play mode the TitleScreen would just disappear because we don't have anything inside the LevelSelect just yet

Adding our level select menu

1. Click the TitleScreen and in the Inspector and uncheck the small tick box at the top (Everything on the canvas should disappear but it's still there)

2. Start by duplicating the PLAY button and calling it BACK, change the text to say BACK too
3. Drag and drop the button into the LevelSelect and it should reappear

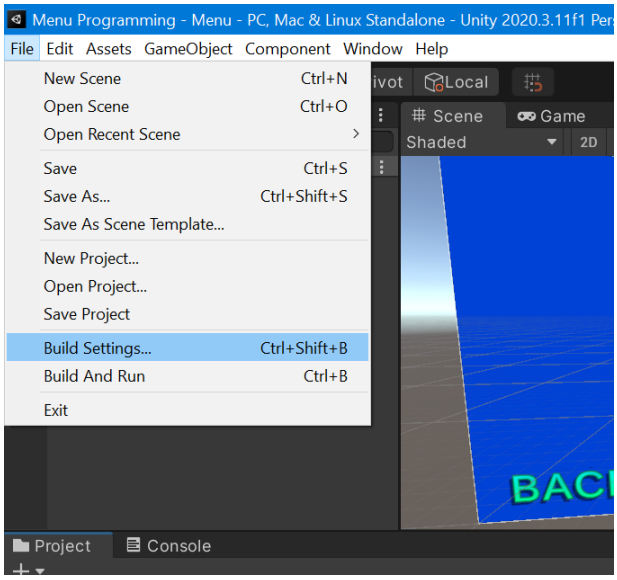


4. Go back to the On Click() in the Inspector for the BACK button
5. Check the first tick box and untick the second

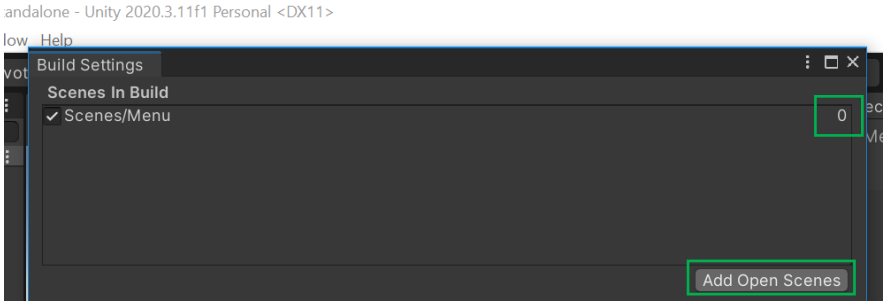
You should now be able to switch between the two menus in Play mode but you will start on the level select, while we are still changing stuff this will not matter but once we are done we will set the TitleScreen as the default.

Making functioning level select buttons

1. Make sure you have other scenes saved for your levels
2. Have your Menu open
3. In the top left corner go File > Build Settings

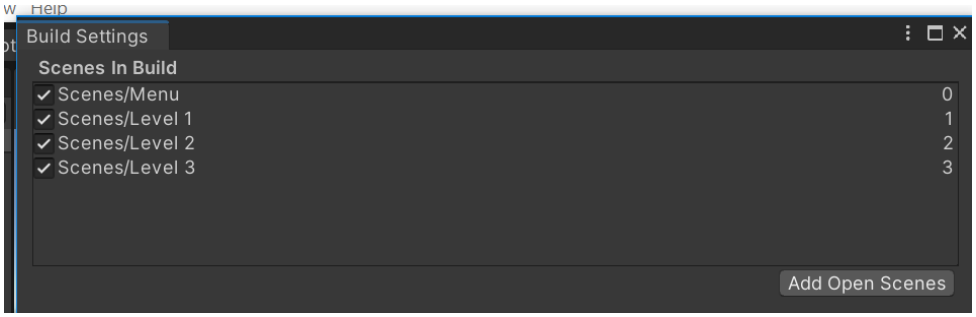


4. Click Add Open Scenes (You will see the menu be added to the list of scenes and it should have a value of 0)

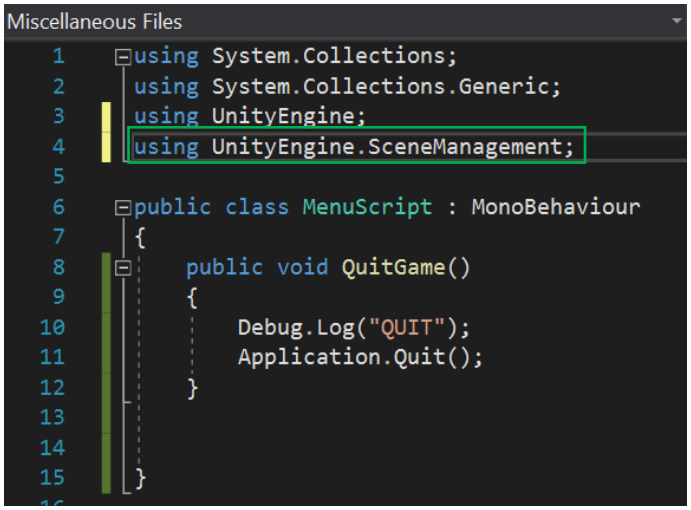


The value of the menu is 0 and for every subsequent level added the number will increase, this number is essentially and ID that you must use whenever you want to access another level. 0 is always called upon first when loading a game so always do your menu before your levels

5. Repeat this for the rest of your levels going in chronological order



6. Open your MenuScript up again and add this line to the top of the code



Since we now want this script to be able to change our scene to a different one so we can access levels we need to include this line

7. Add this function and save your script

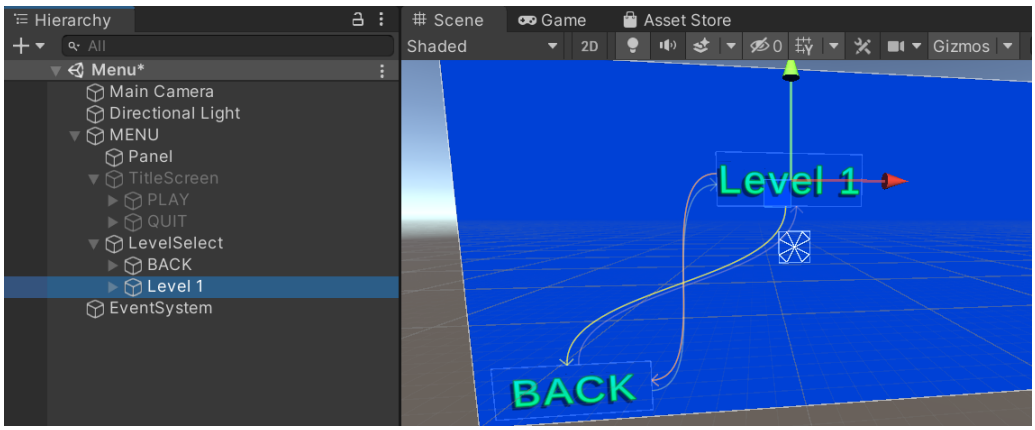
```
MenuScript.cs*
Miscellaneous Files
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4   using UnityEngine.SceneManagement;
5
6   public class MenuScript : MonoBehaviour
7   {
8
9       public int index;
10
11      public void LevelSelect()
12      {
13          Debug.Log("Level Change");
14          SceneManager.LoadScene(index);
15      }
16
17      public void QuitGame()
18      {
19          Debug.Log("QUIT");
20          Application.Quit();
21      }
22
```

In Unity you are able to hard code in what level you go to by assigning the Scene ID mentioned earlier however instead of that we have put a public int in its place.

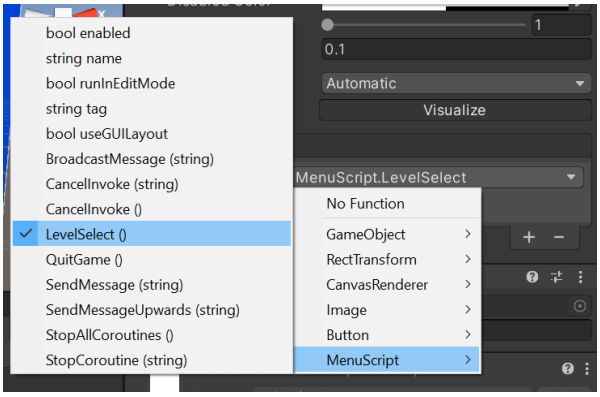
This means that when we set the function on our buttons we simply change the level number which is more efficient than changing the function.

When dealing with scene switching this is a good habit to get into as it results in less code and means code can be reused more easily

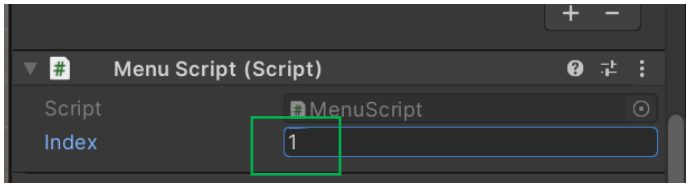
8. Return to Unity and duplicate the QUIT button and rename it Level 1 (change the text too)
9. Move this button into the LevelSelect



10. Go to the On Click() section in the hierarchy
11. Where it says MenuScript.QuitGame click and navigate to the function we just made by going MenuScript > LevelSelect



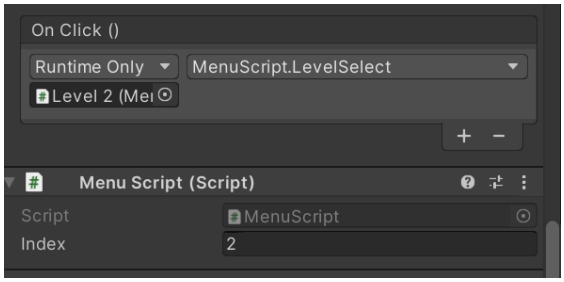
12. Go to where the script is in the Inspector and change the number on index to 1



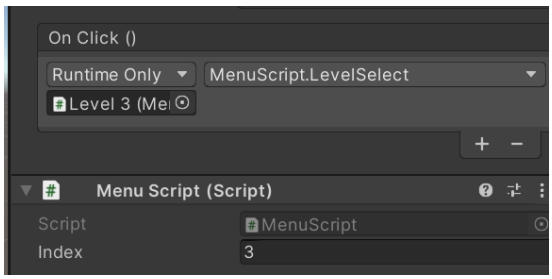
The reason we set it to 1 is because that was the ID that level 1 had, if the level had a different ID we would change it to its respective n number which is what we are going to do for the level 2 and 3 buttons

13. Check in play mode to see if the button works

14. Duplicate the Level 1 button twice to create the Level 2 button
15. Make sure the Level 2 button is in the On Click() part in the inspector and the level change function is assigned
16. Change the index on the script in the inspector to the ID for level 2 (Check the build settings if you need a reminder)

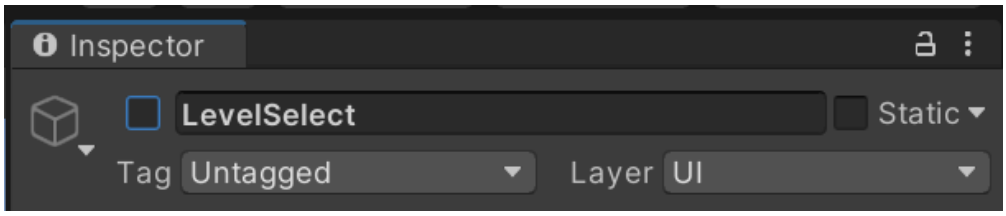


17. Repeat for the level 3 button

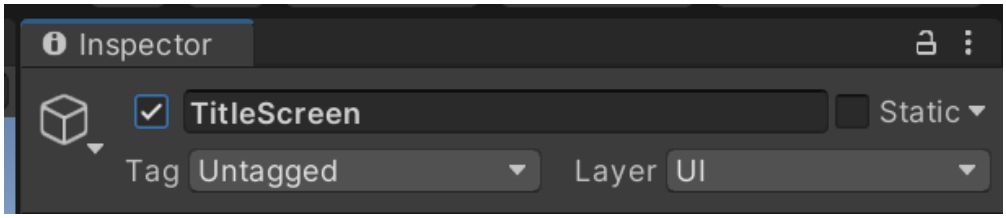


Assigning the title screen as the default

1. Go to the LevelSelect and untick the box at the very top of the inspector



2. Go to the TitleScreen and tick the box at the top



That is how to make a basic main menu for your game, using more buttons you can make other sub menus like an options menu or a credits section. By using other objects like sliders, raw images you can make your UI more interesting or by making use of a camera and placing objects around it you can change the theme of your menu.

These are some menu's I recently made that use the raw images and background objects/lighting