AGA206 Roll A Ball
Mandatory Modules

Setup before any module

1. Change the Tag on the Player Game Object (Ball) to “Player”

2. Setup a tag “Ground” and put it on all the ground piece prefabs and anything else we want to have player control on (eg Ramps)

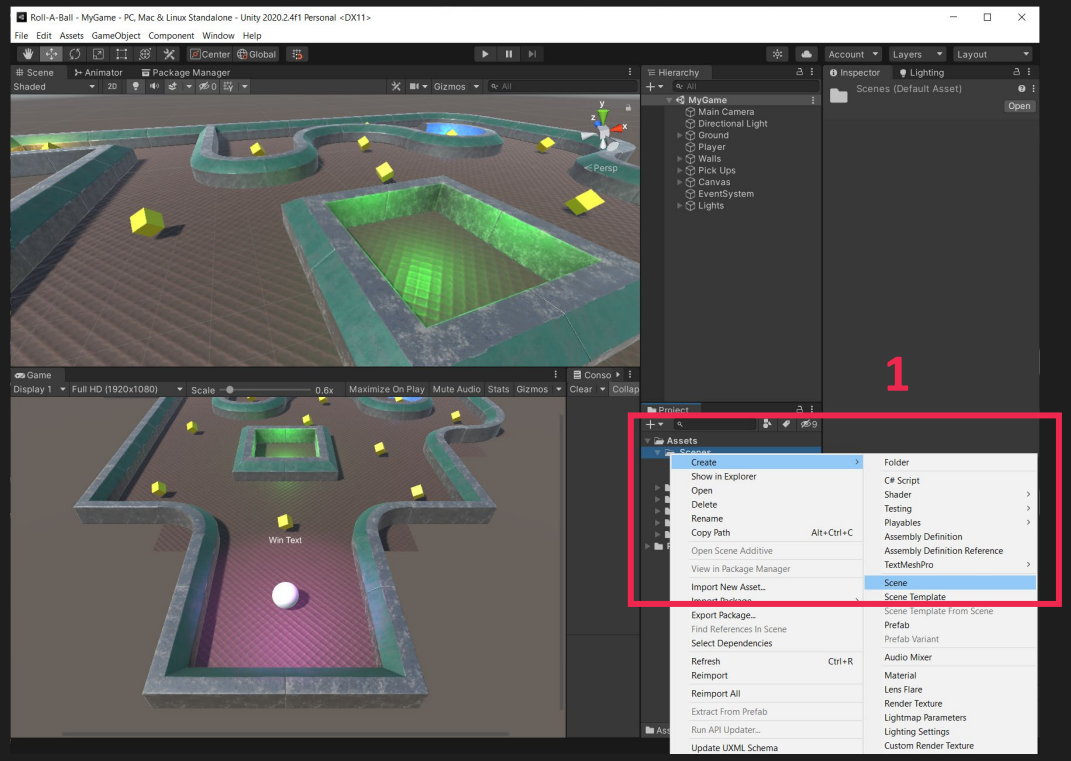
3. Make the tag “Wall” and apply to all the wall prefabs

4. Turn our wall meshes colliders to not be convex

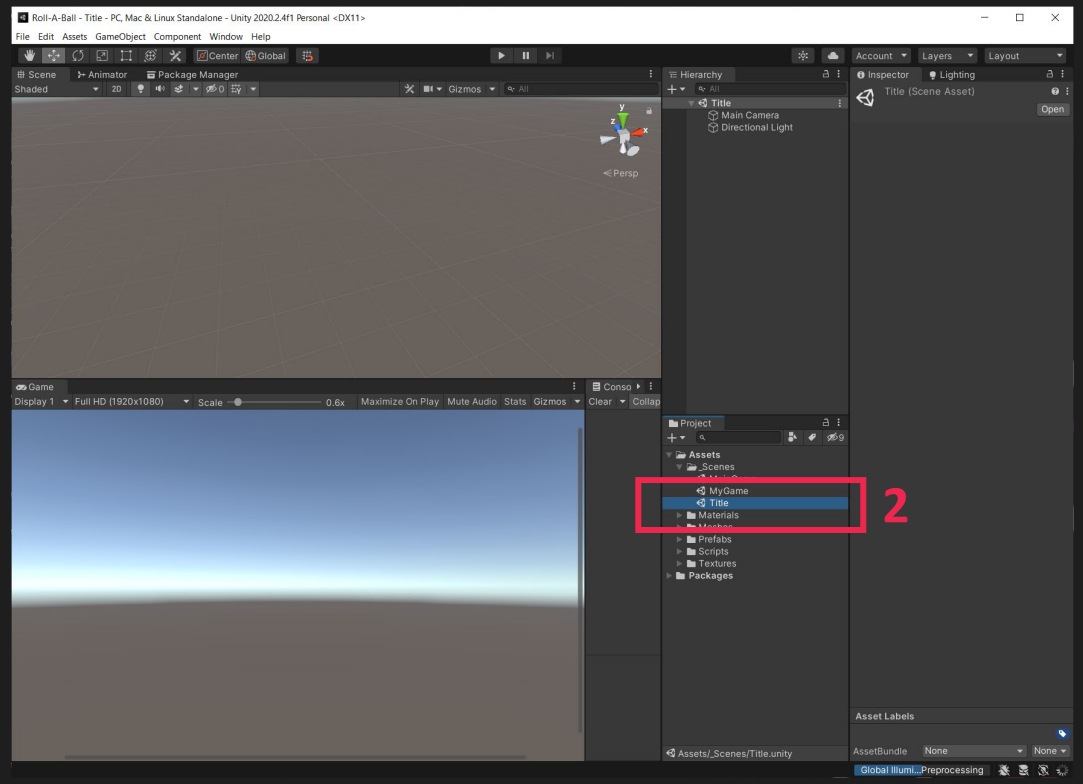
1. Start and Game Over Screen	0pts
Pre-requisite: None	
Create a title screen that has two buttons - one to start the game, one to quit the game. When the player clicks the start game button, the game begins. When the player clicks the quit button, the game closes. When the player collects all the pickups (or any other game over scenario), a win screen is shown with a button to return to title. If the player clicks on the return button, they go to the title screen.	
Design: UI Layout	
Programming: UI Screens and Buttons, Scene Management	

Create the title scene

1. Right click on the _Scenes folder then go to Create > Scene

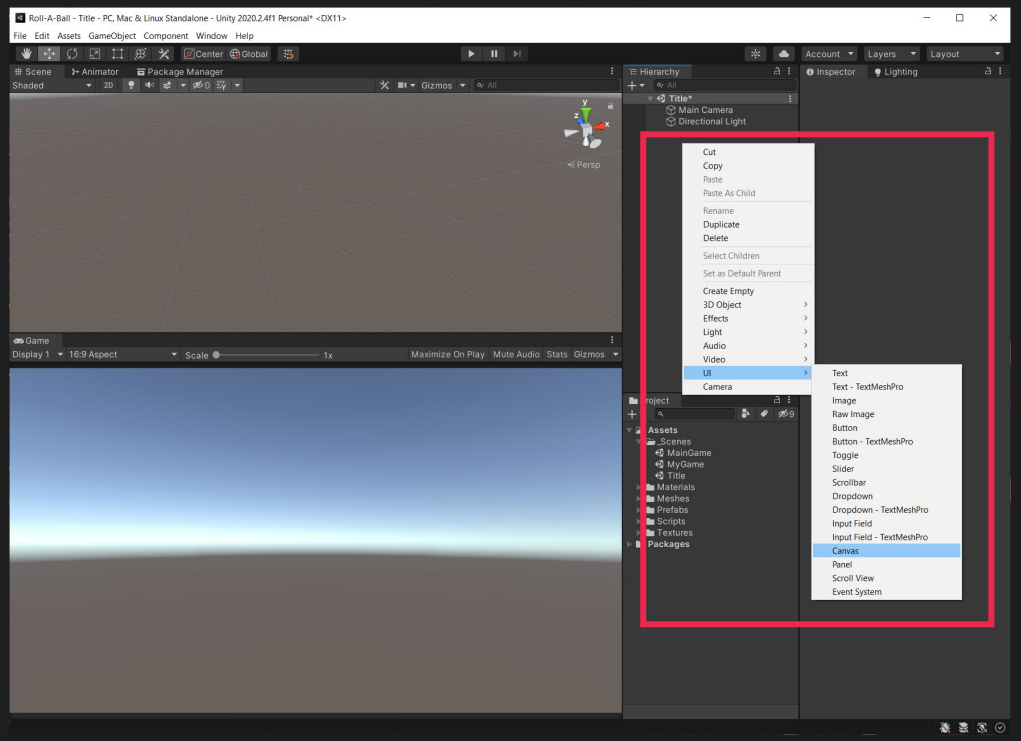


2. Name the scene ‘Title’ and double click on it to enter the scene

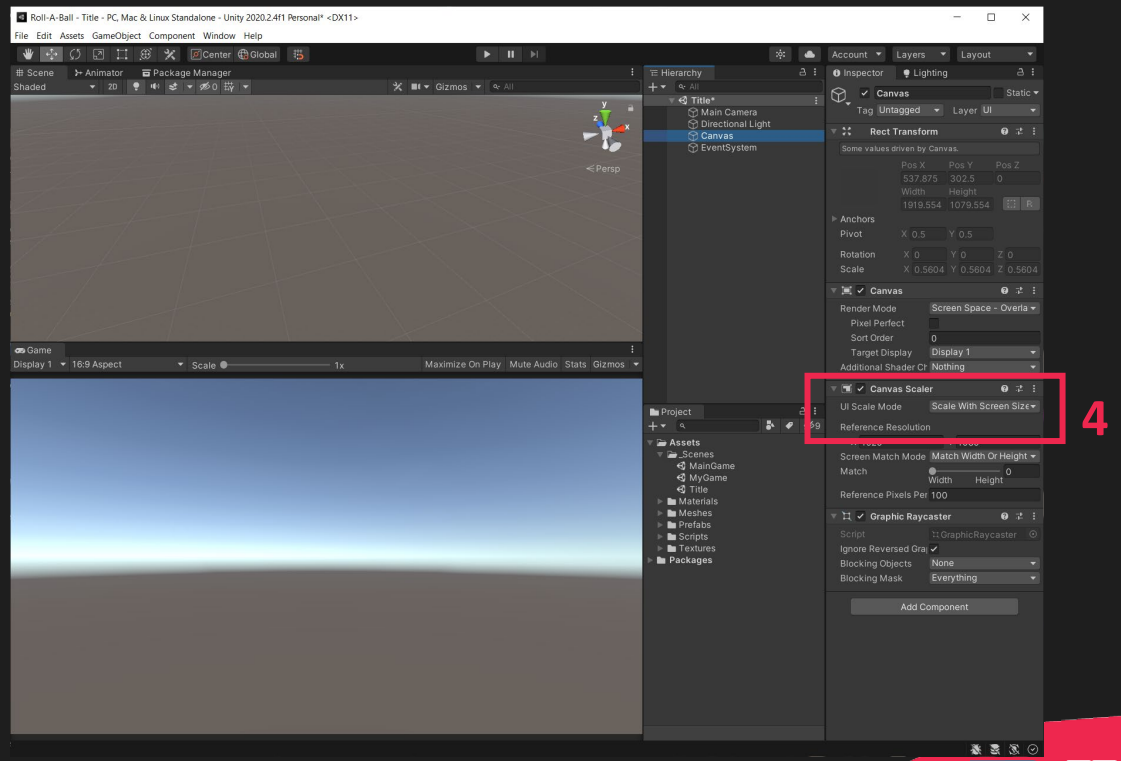


Setup the canvas

3. In the hierarchy, right click and go to UI > Canvas

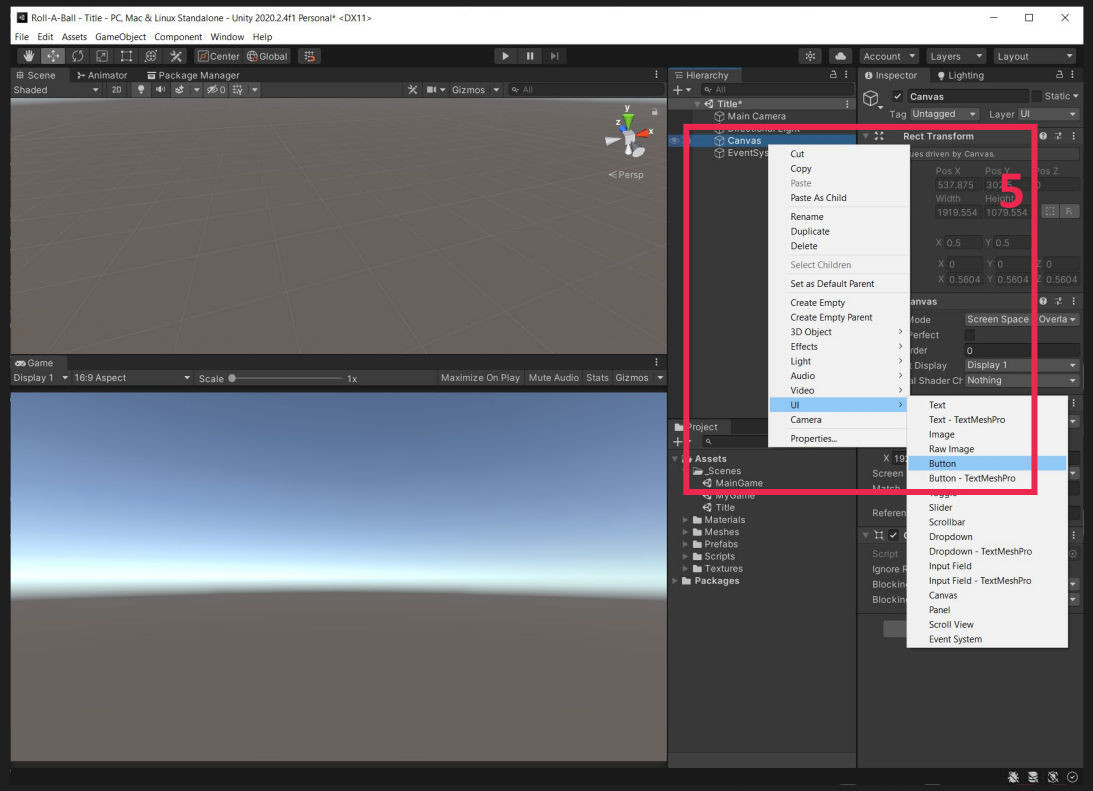


4. On the Canvas Scaler component, change the settings as in the image:
   - UI Scale Mode : Scale with screen size

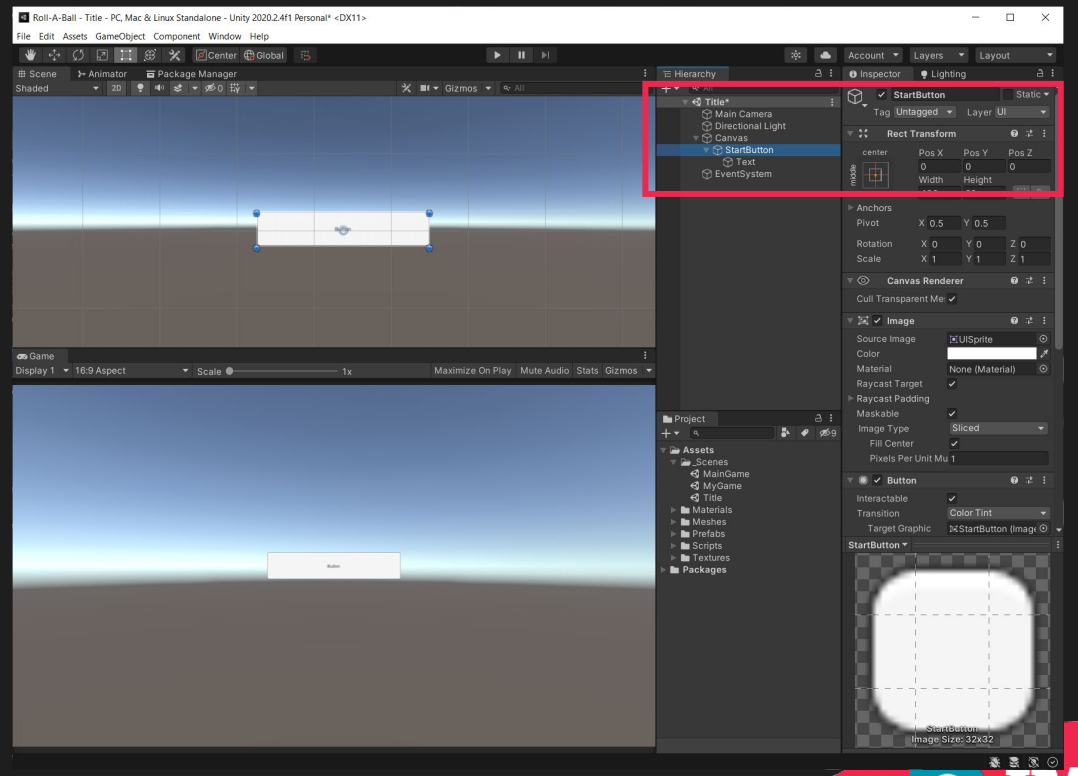


Setup the button

5. In the hierarchy, right click on the Canvas and go to UI > Button



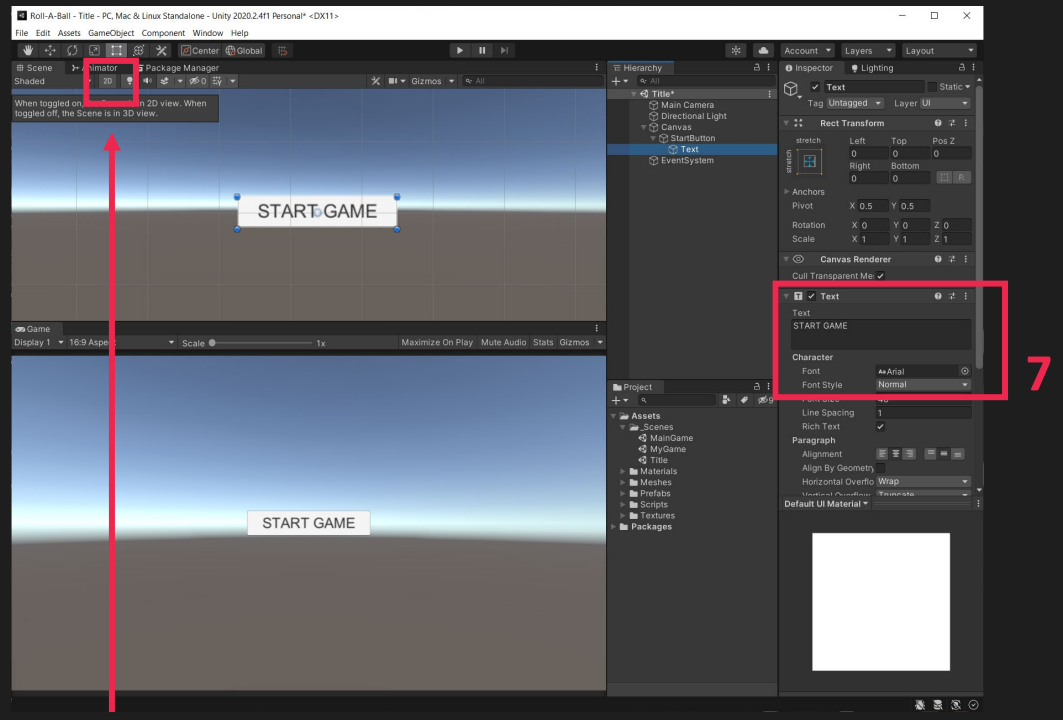
6. Rename the Button to be StartButton, and set the properties on the button as follows:
   - Pos X = 0, Pos Y = 0, Pos Z = 0



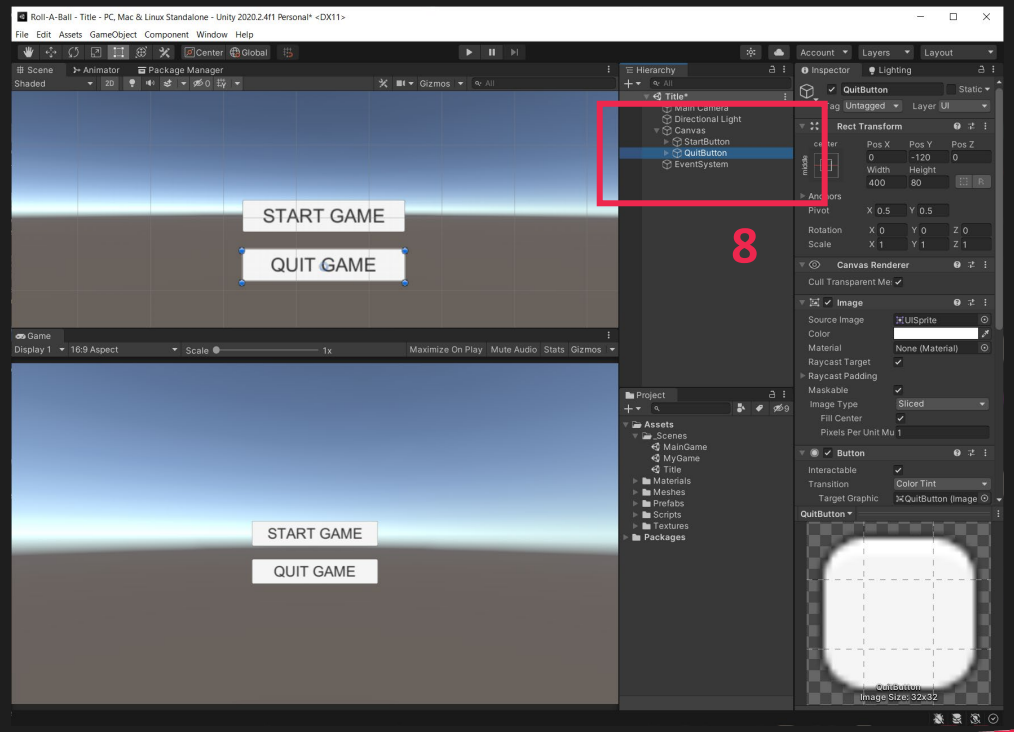
Setup the buttons

7. Click on the Text object in the StartButton, change the text to START GAME

8. Duplicate the StartButton, move it down below the StartButton, rename it to QuitButton, and change the text to QUIT GAME

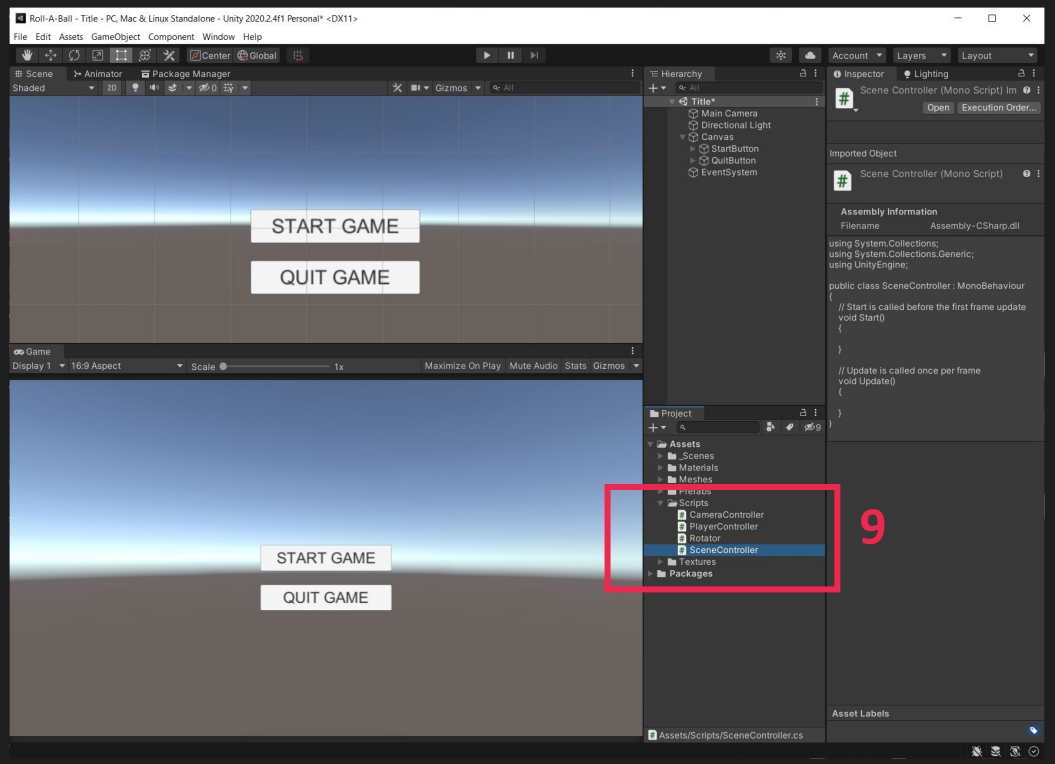


When doing UI, you can change the viewport to 2D to make it easier to position things



Creating the SceneController

9. In our Scripts folder, create a C# script called SceneController (no spaces!)



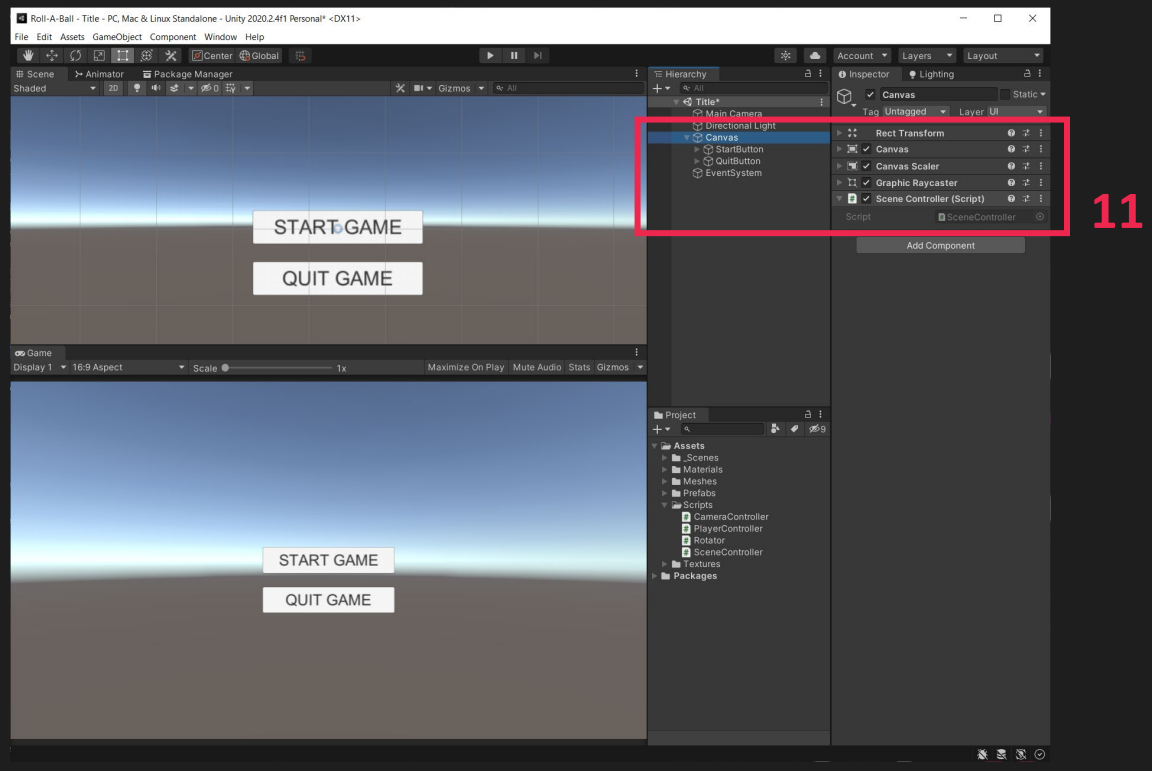
10. Edit the script to be exactly as in the image

```
using UnityEngine;
using UnityEngine.SceneManagement;

public class SceneController : MonoBehaviour
{
    //Will change our scene to the string passed in
    public void ChangeScene(string _sceneName)
    {
        SceneManager.LoadScene(_sceneName);
    }

    //Reloads the current scene we are in
    public void ReloadScene()
    {
        SceneManager.LoadScene(SceneManager.GetActiveScene().name);
    }

    //Loads out Title scene. Must be called Title exactly
    public void ToTitleScene()
    {
        SceneManager.LoadScene("Title");
    }

    //Gets our active scenes name
    public string GetSceneName()
    {
        return SceneManager.GetActiveScene().name;
    }

    //Quits our game
    public void QuitGame()
    {
        Application.Quit();
    }
}
```

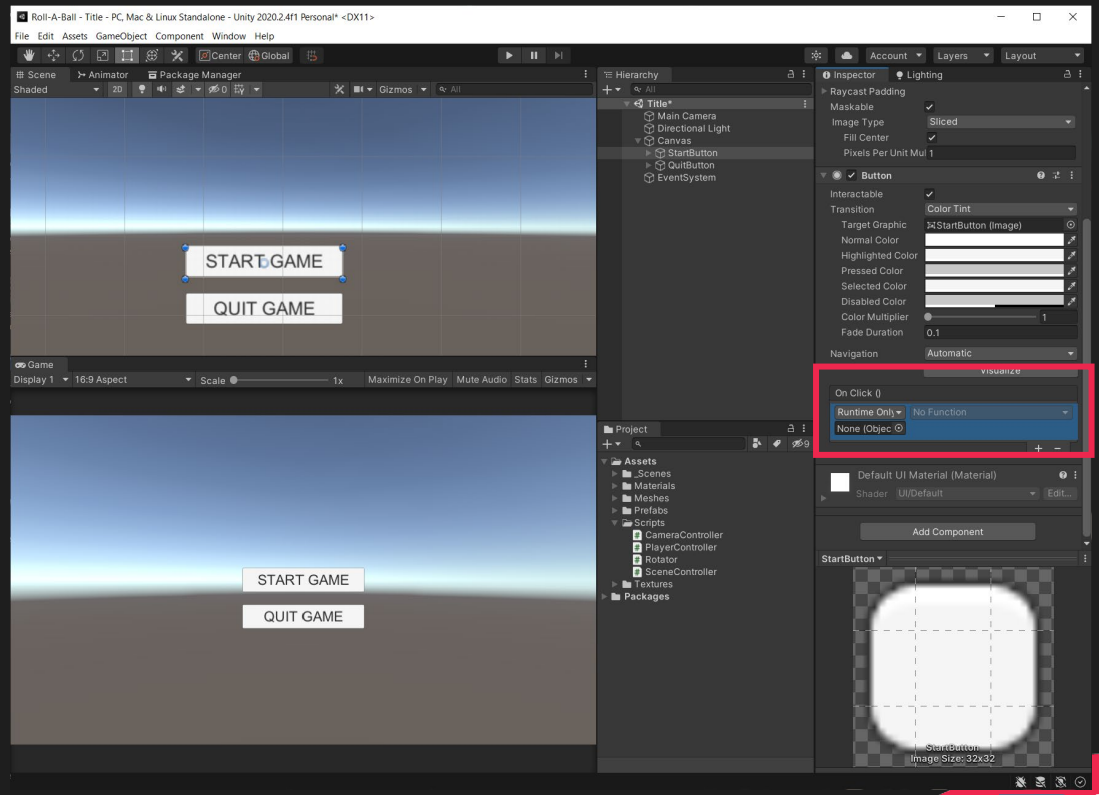
10

Button Click Functionality

11. Back in Unity, drag the SceneController script onto the Canvas Game Object

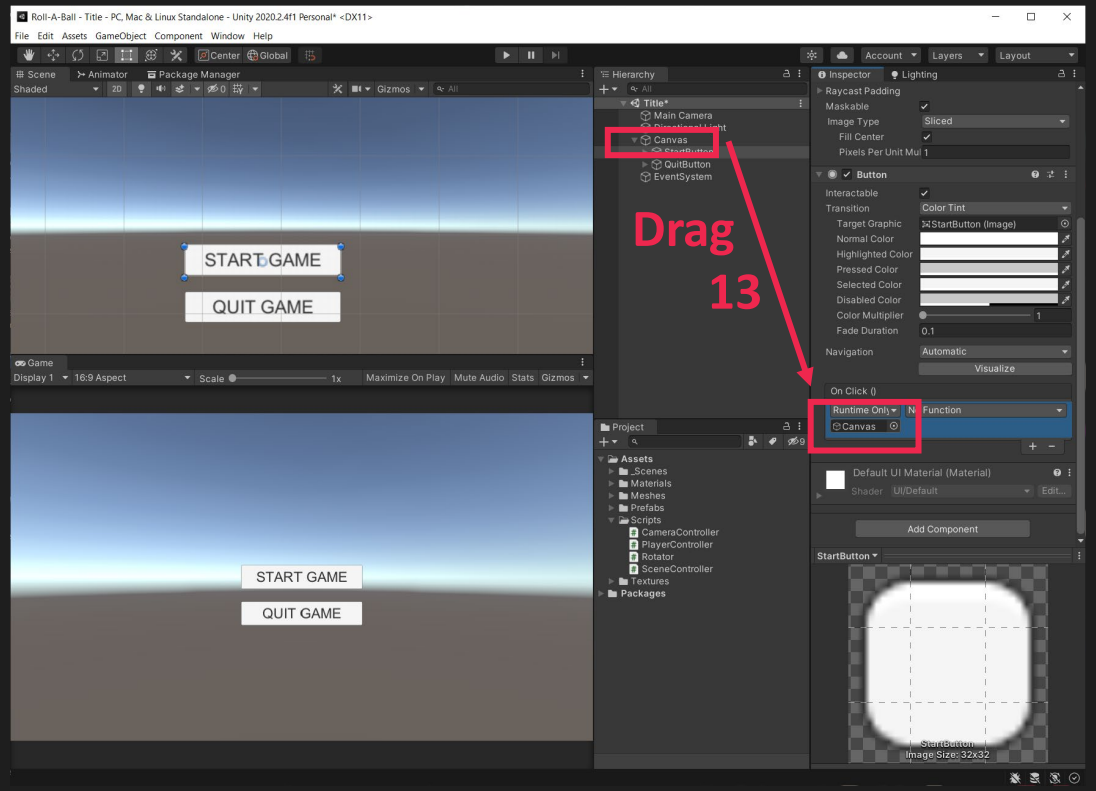


12. Click on the StartButton, then scroll to the OnClick() part of the Button component. Click on the + icon to add a click event

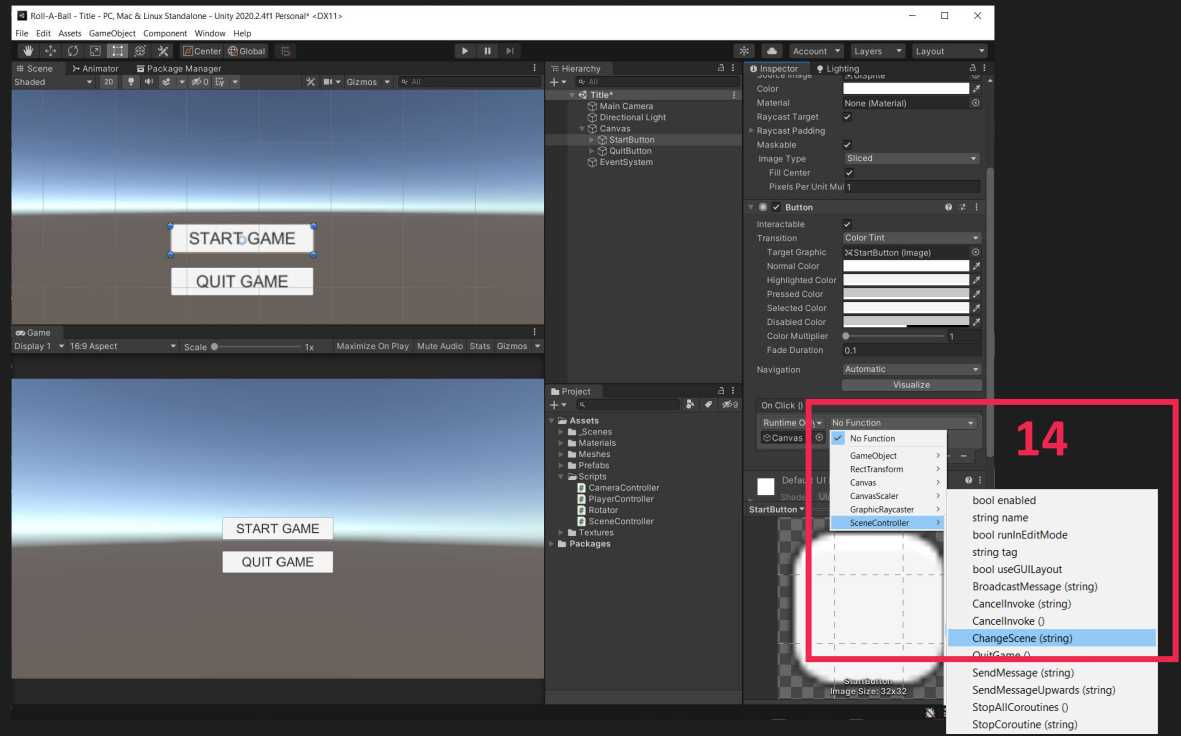


Button Click Functionality

13. Click drag on the Canvas Game Object and release it on the box in the buttons On Click section under the Runtime Only dropdown

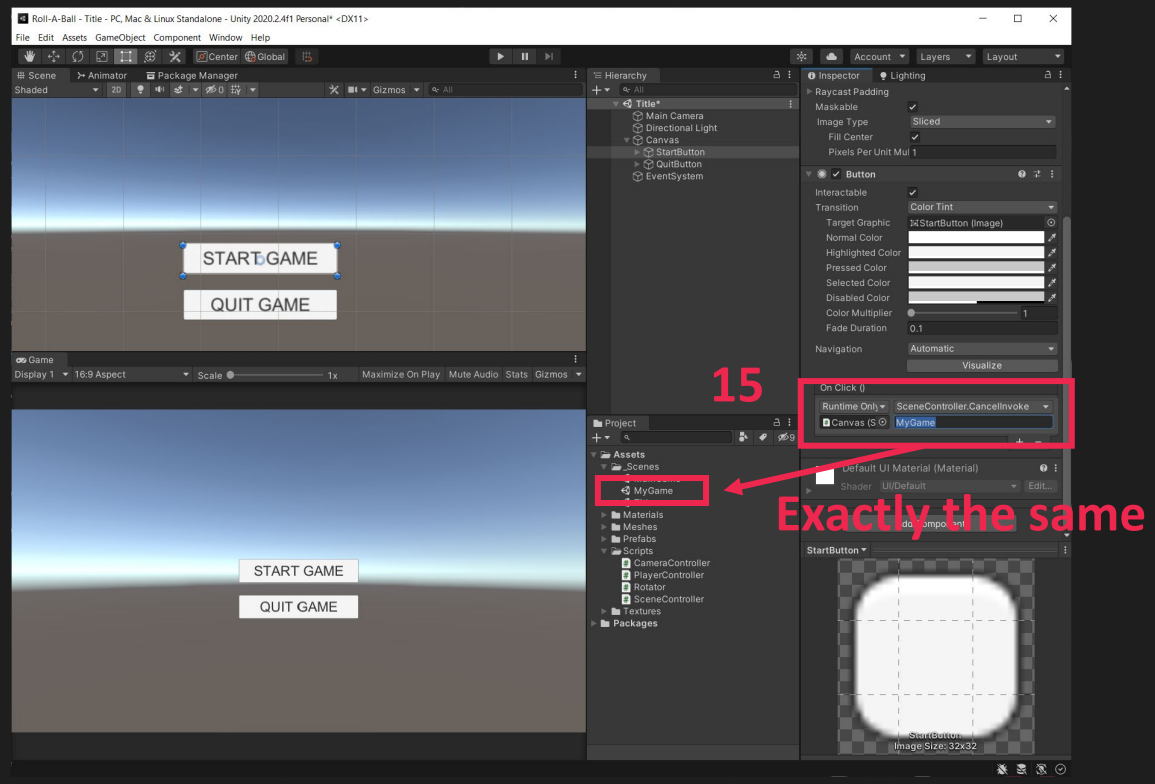


14. In the Function dropdown, scroll to SceneController > ChangeScene(string) and select this option

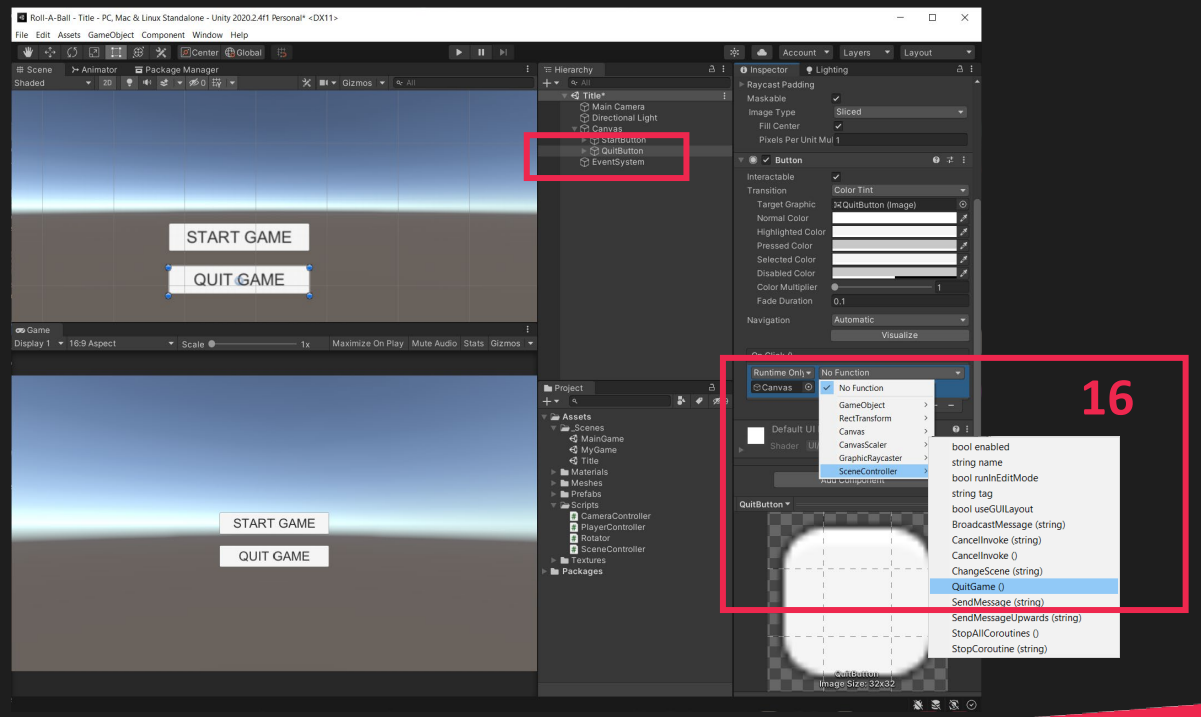


Button Click Functionality

15. In the new space in the On Click section, write the name of the scene you want to load EXACTLY as it appears in the project folder
    - NOTE: Your scene name may be different to the one in the picture so adjust accordingly



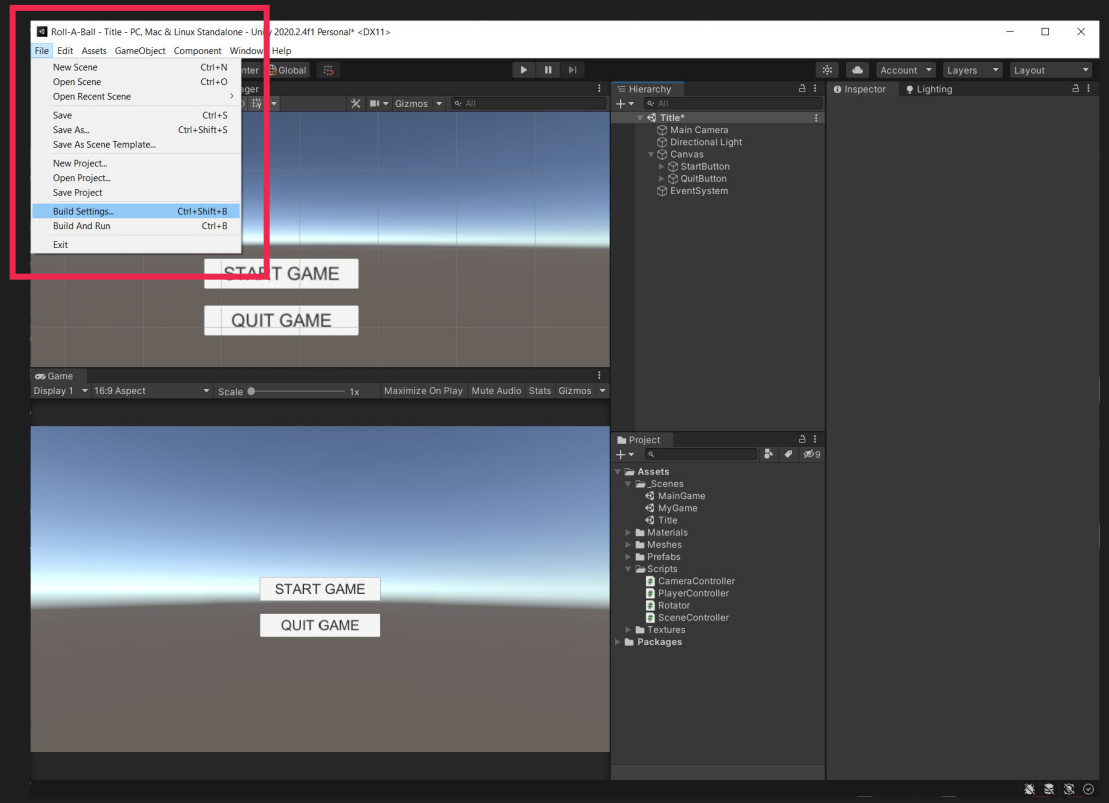
16. Repeat steps 12 - 13 on the QuitButton but change the dropdown function to QuitGame()



Adding the scenes to the build

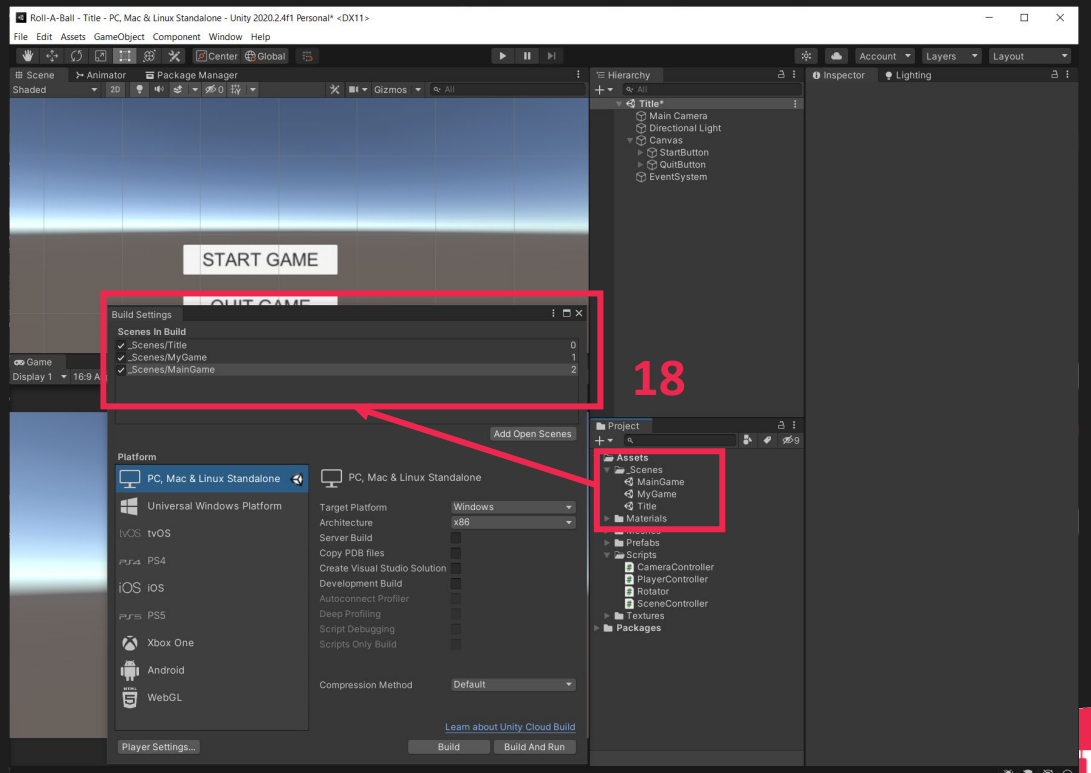
17. Go to the build settings in the File menu

17



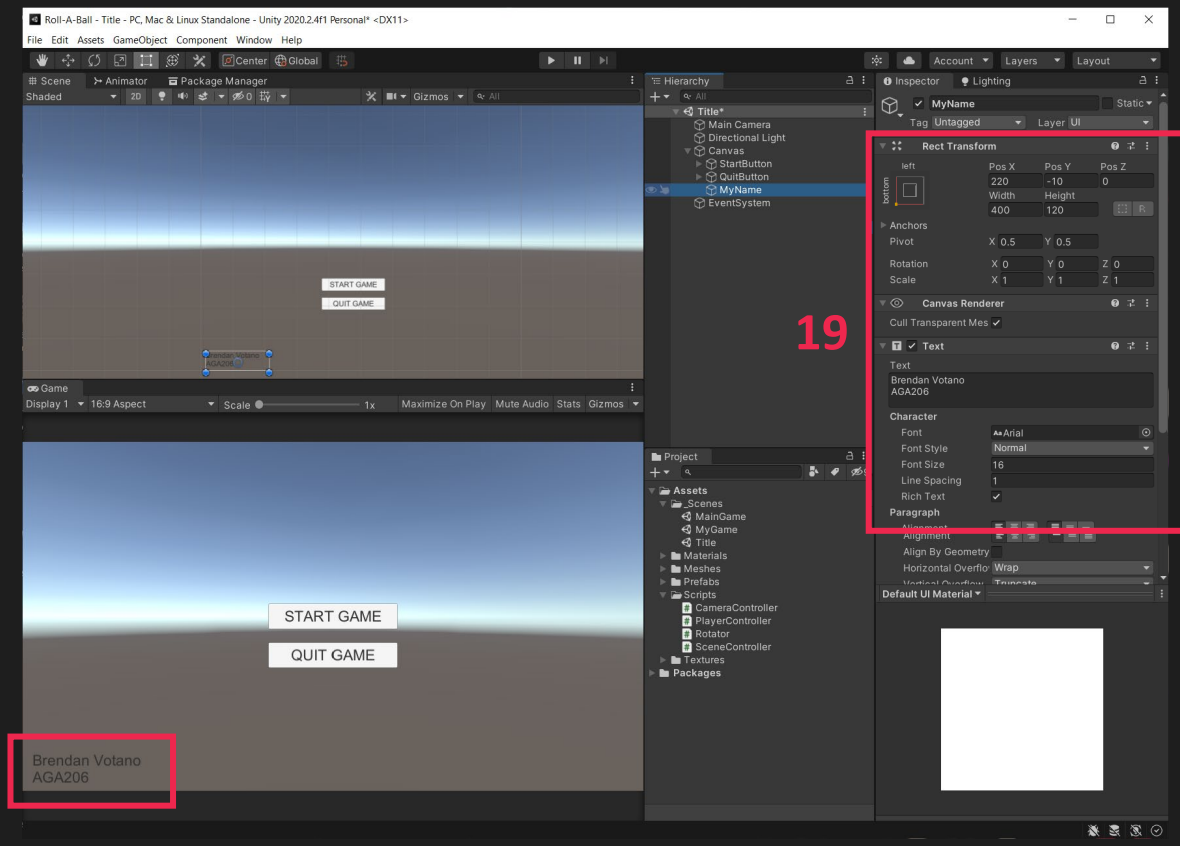
18. In the Scenes In Build, click drag the scenes you want in the build from the Project folder.
    - Whenever you add a new scene to your game, you should also add it to the Scenes In Build otherwise it will not be included in the build.

18

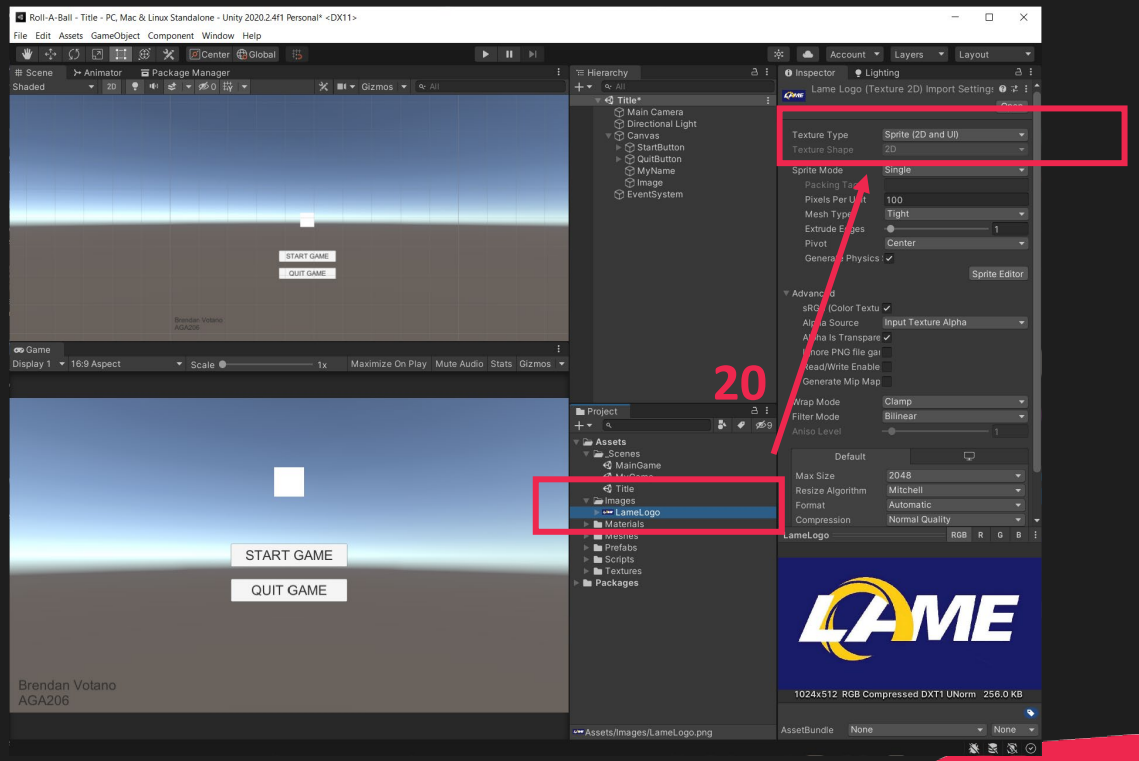


Adding the final touches

19. Create a UI Text Element on the Canvas (UI > Text), call it MyName, position it in the bottom corner and put in your name and AGA206.
    - Copy the settings below to get it looking right



20. Add a UI Image to the canvas.
    Any images we want to show on the UI, we should Image's Texture Type to Sprite.

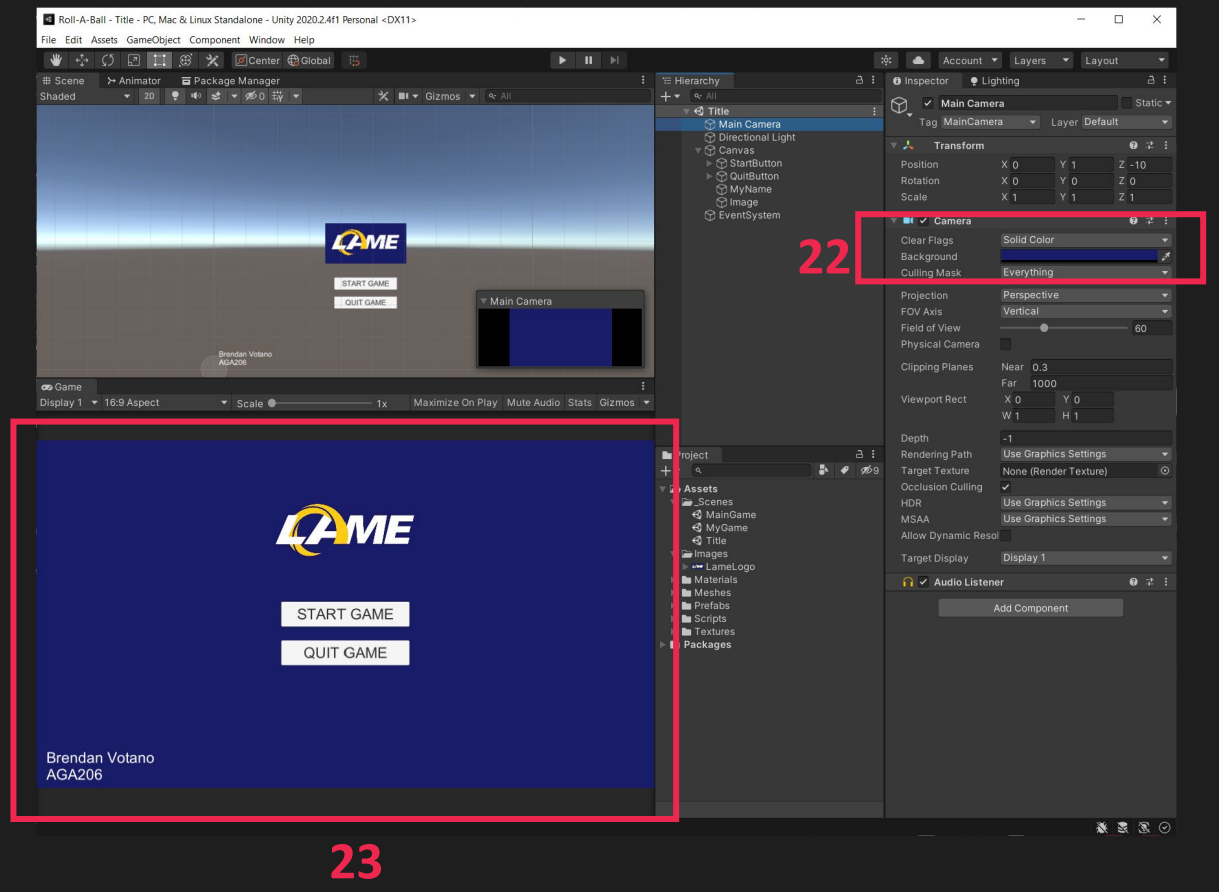


Adding the final touches

21. Drag the image from the Project Folder to the UI Image component.
    Resize the image as needed

22. Change the background to a flat colour by changing the Camera Clear Flag and Background settings

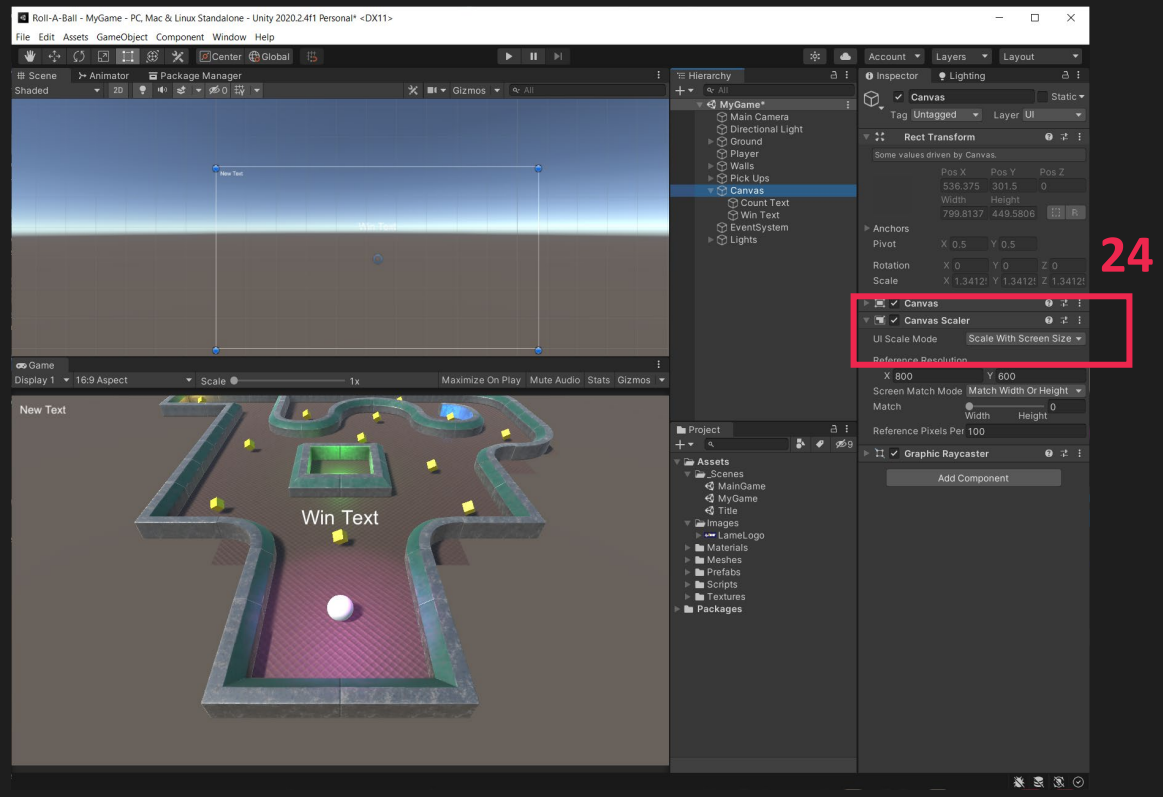
23. Adjust anything else to make your title screen look awesome!
    - Text colours
    - Button colours
    - Etc.



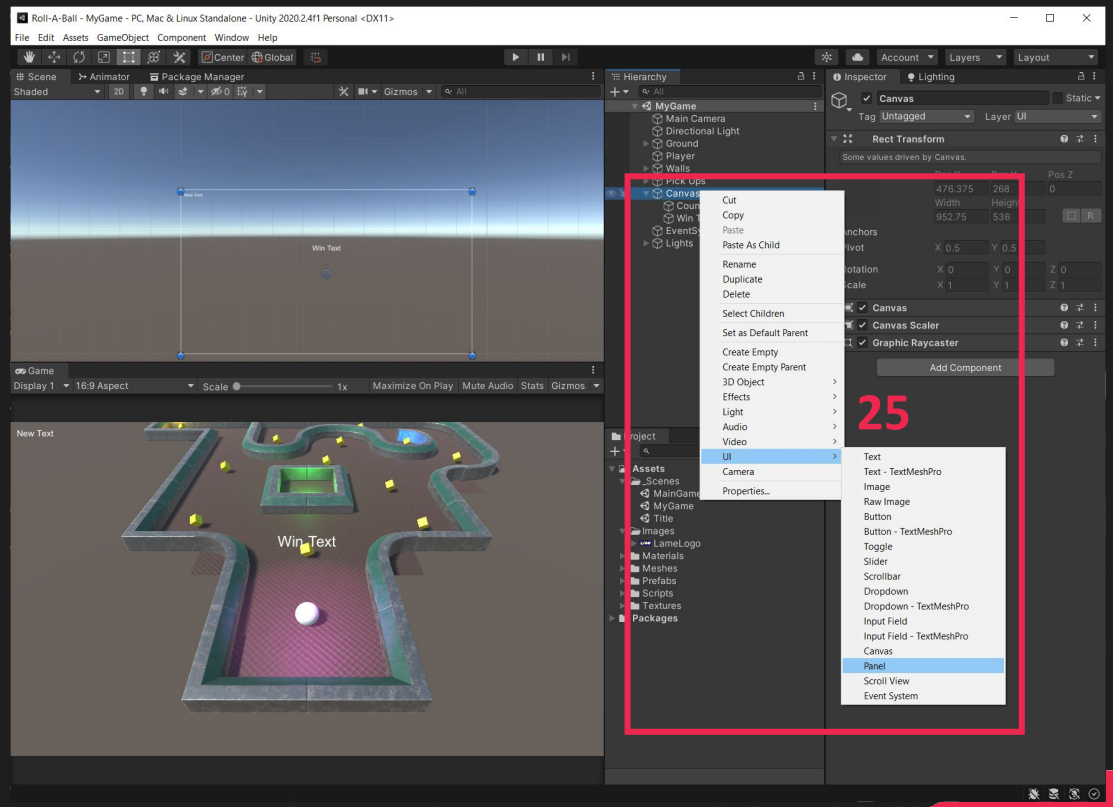
- Other modules will add more buttons to this title screen so you may want to revisit these notes again when you need to

Game over screen in game

24. Back in our Game Scene, click on the Canvas and change our Canvas Scaler component UI Scale Mode to Scale With Screen Size

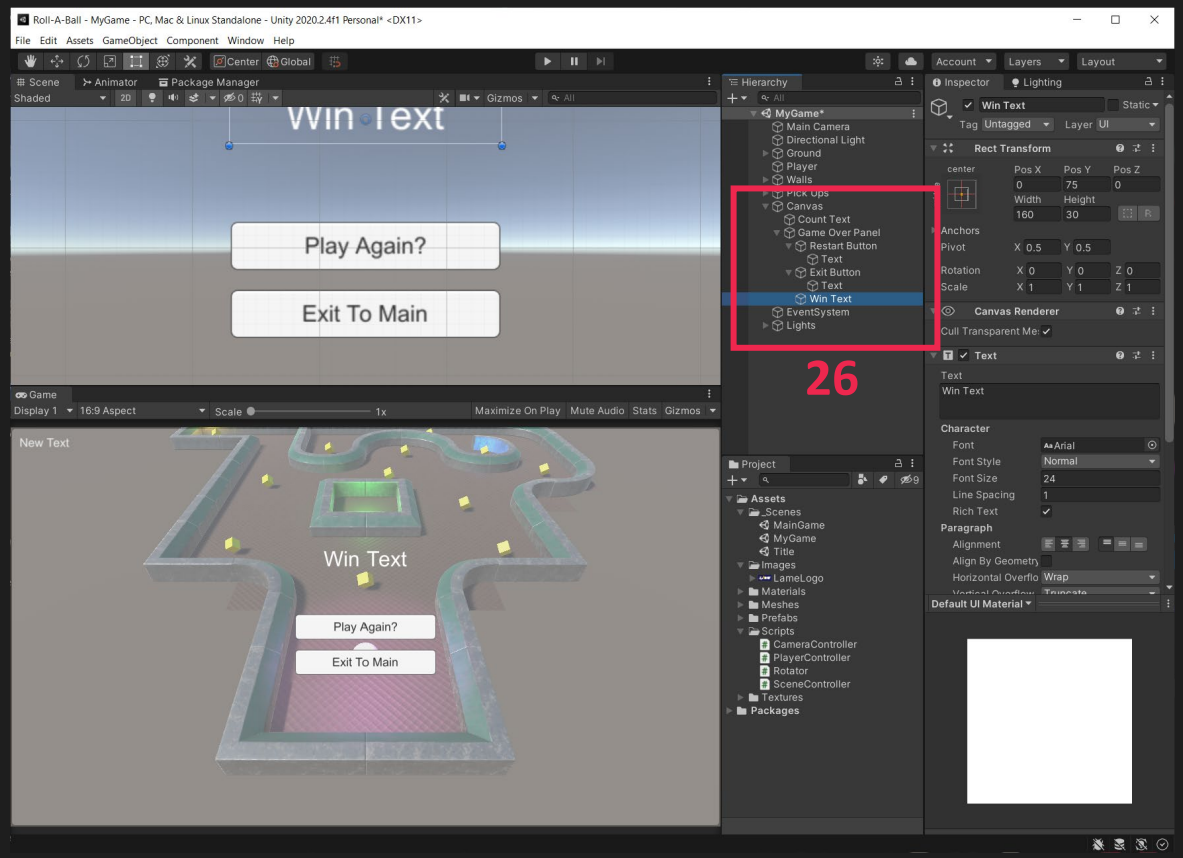


25. Add a UI Panel to the Canvas and rename it to Game Over Panel

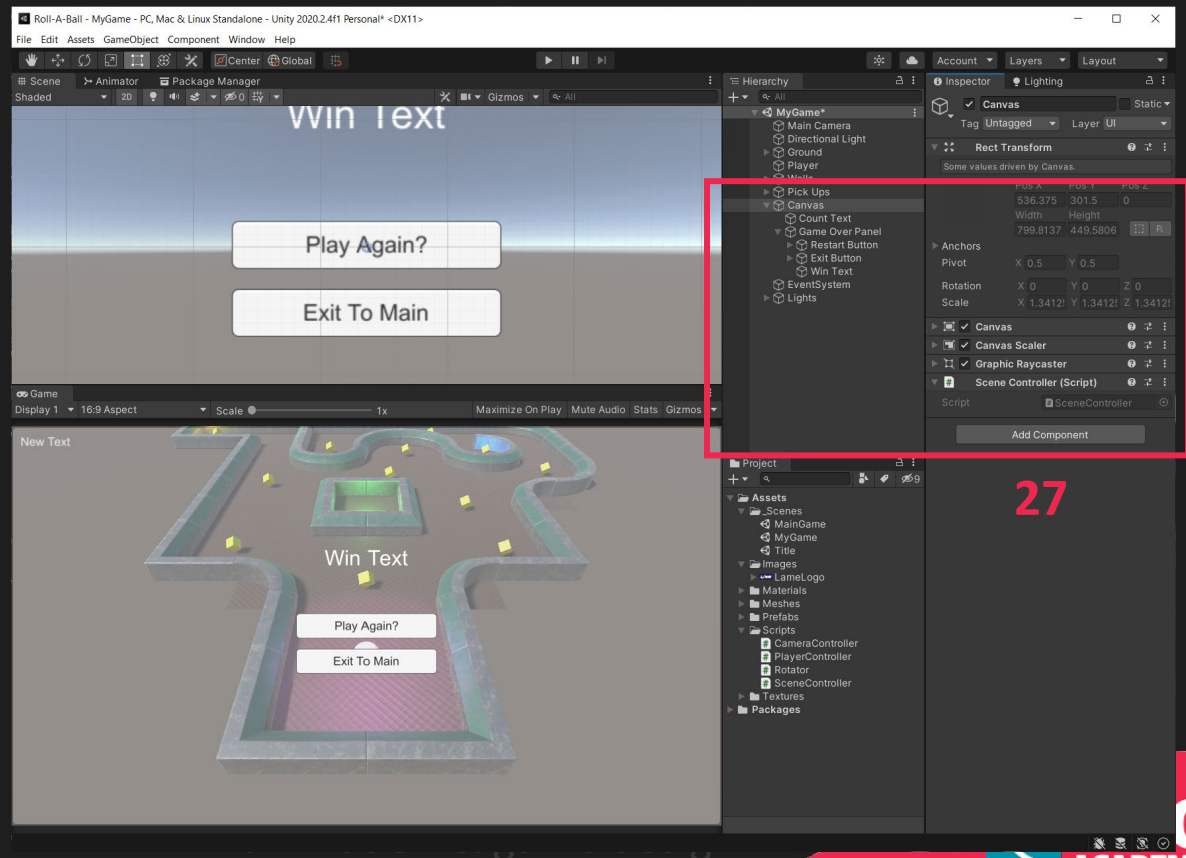


Create the buttons

26. Create two buttons in the Panel.
    Name them Restart Button and Exit Button.
    Size and position them as you see fit
    Click drag the Win Text into the Game Over Panel



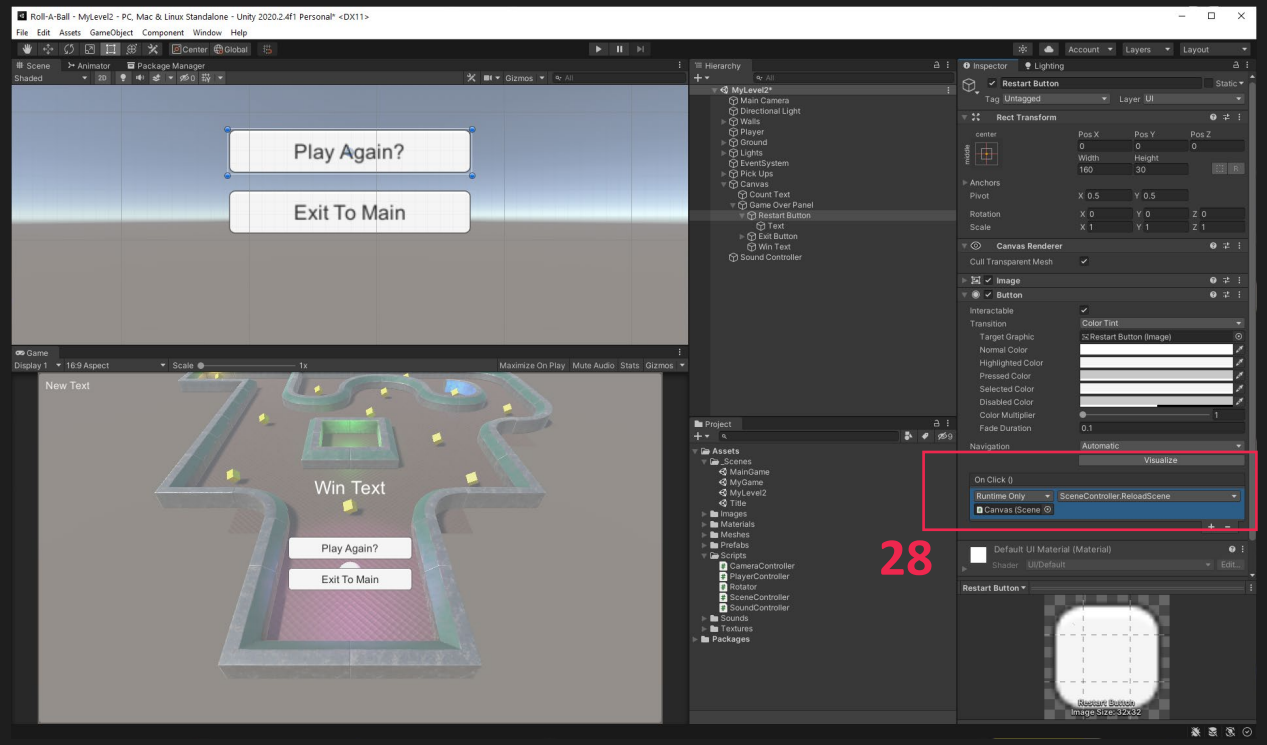
27. Add the SceneController script to our Canvas Game Object

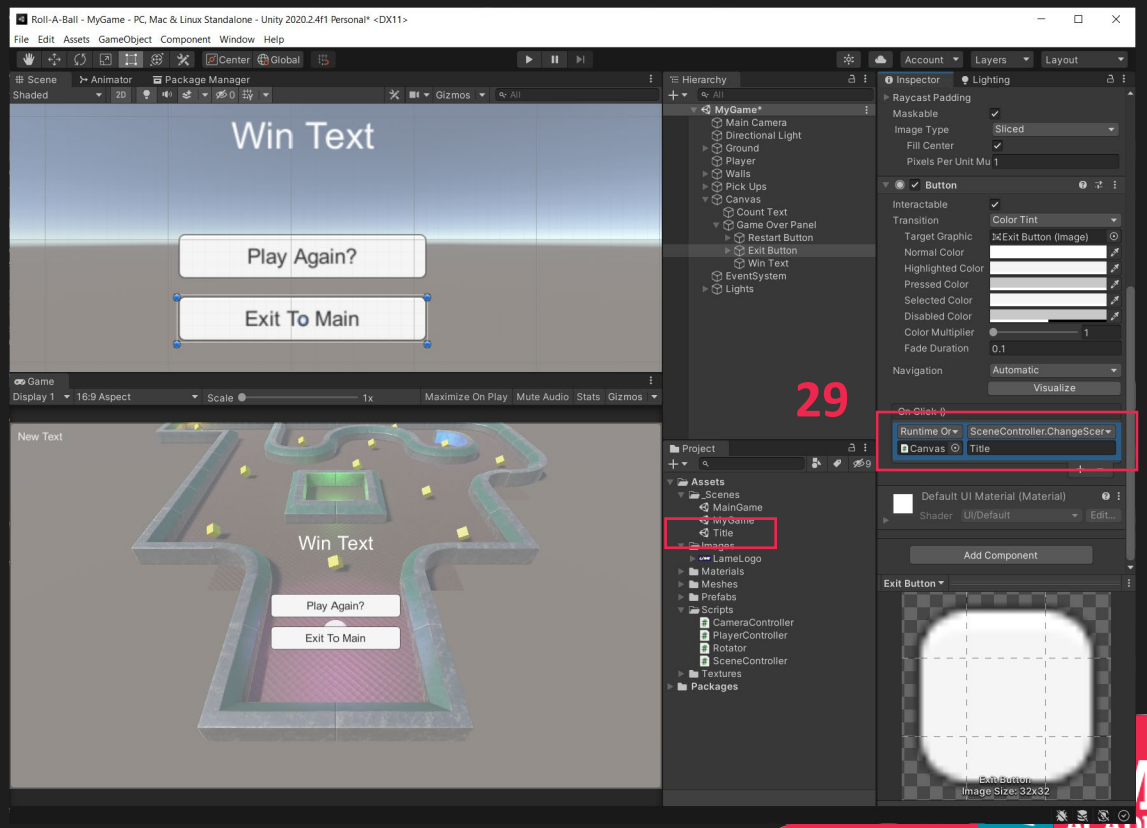


Hook up the buttons

28. Create the Restart Button On Click event as we did in steps 12 – 14, but change the dropdown function to ReloadScene

29. Do the same again for the Exit Button, but put the name of your title scene into the field
    - In this case it's called Title





Modifying the PlayerController

30. In the PlayerController script, add/modify the following highlighted sections
    - NOTE: There is other code between these sections that has been omitted to save space

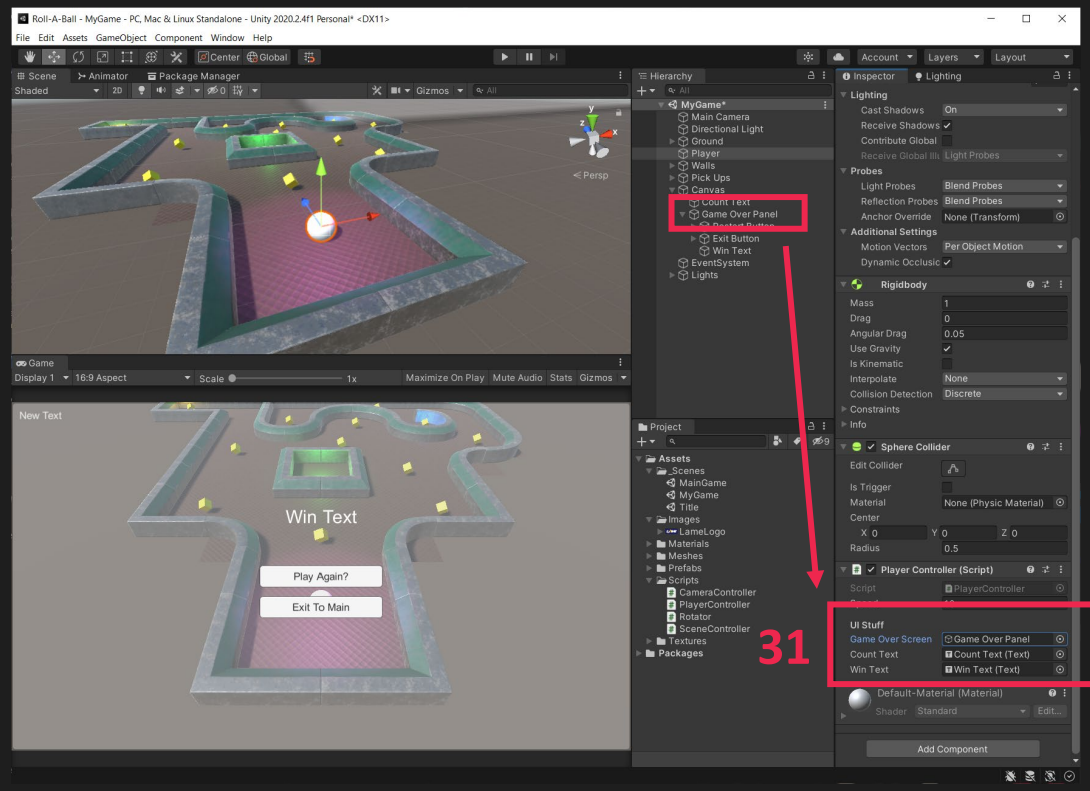
```
public class PlayerController : MonoBehaviour
{
    public float speed = 10.0f;

    [Header("UI Stuff")]
    public GameObject gameOverScreen;
    public Text countText;
    public Text winText;

    void Start()
    {
        rb = GetComponent<Rigidbody>();
        count = 0;
        gameOverScreen.SetActive(false);
        SetCountText();
        winText.text = "";
    }
```

Change to this →

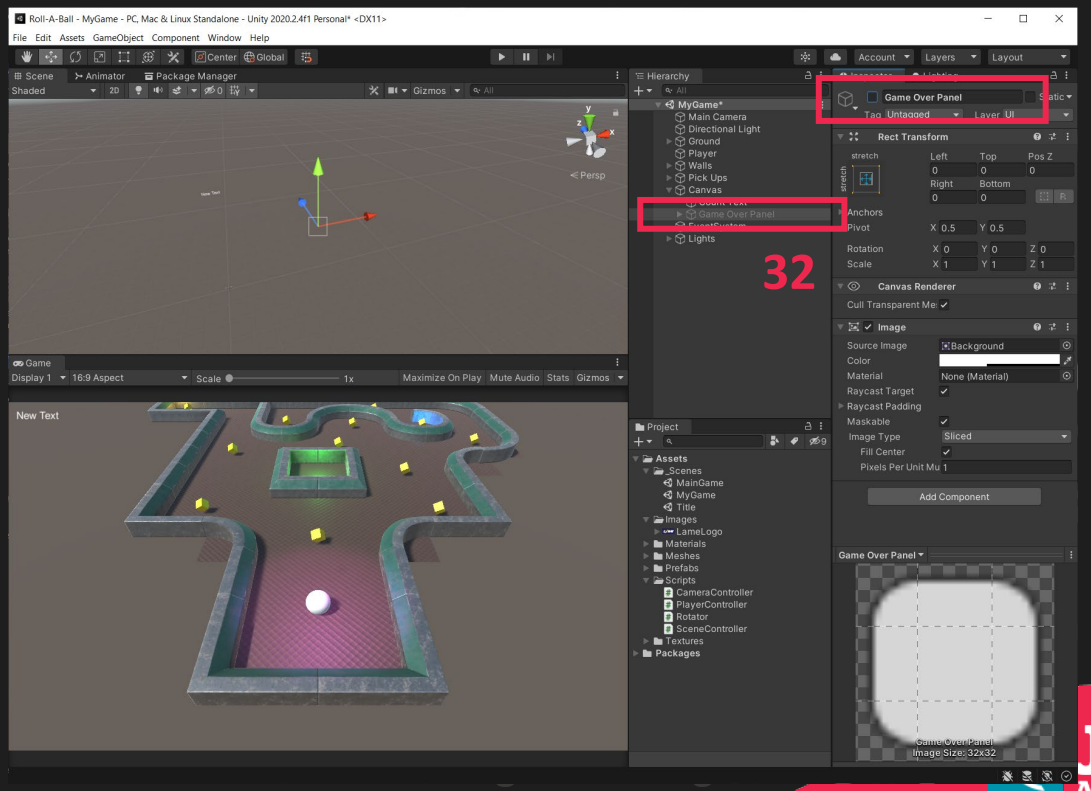
```
void SetCountText()
{
    countText.text = "Count: " + count.ToString();
    if(count >= 12)
    {
        WinGame();
    }
}

void WinGame()
{
    gameOverScreen.SetActive(true);
    winText.text = "You Win!";
}
```

Hook up to the Player

31. Click on the Player, then drag the Game Over Panel to the Game Over Screen variable on the PlayerController script



32. Now that we have finished the Game Over Screen, we can toggle it off so we can see the game world better when working on it. Then toggle on the Game Over Panel when we want to change it



DONE!!!!!

You should now have a functioning Title Screen

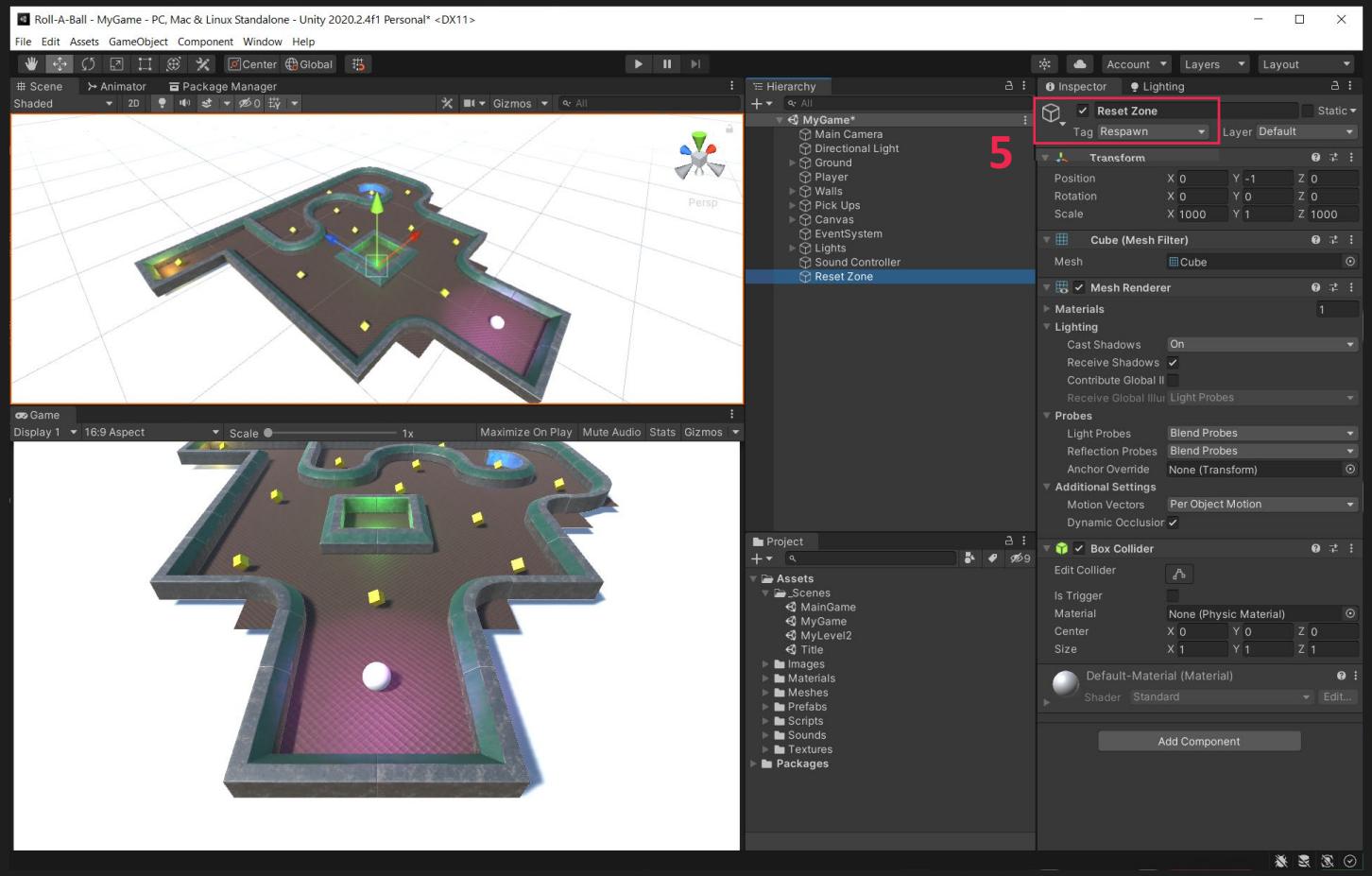
And a functioning Game Over Screen

Feel free to customise them to make them a little more exciting such as changing colours, fonts and images

2. Reset Zone	0pts
Pre-requisite: None	
Reset the player when they hit a specified collider	
Design: Reset Zone Material	
Programming: OnCollisionEnter, Lerping	

Setup the Zone

1. Create a 3D Object > Cube and zero it out
2. Rename the cube to Reset Zone
3. Scale it to x = 1000, y = 1, z = 1000
4. Position it on the y to -1
5. Add the tag Respawn to the object



Setup the Material

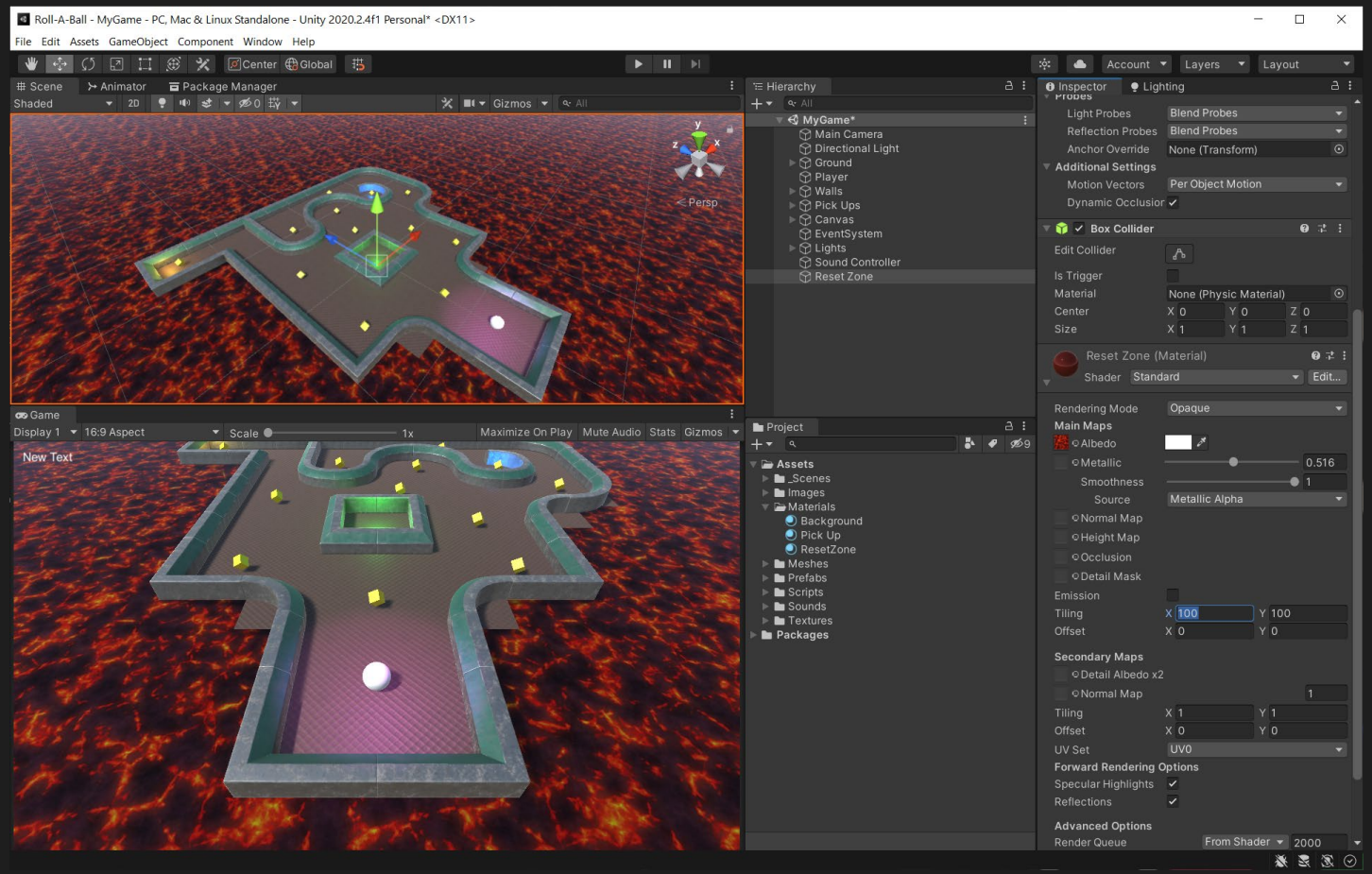
6. Create a new Material in the Materials folder called ResetZone

7. Drag this new material onto the Reset Zone

8. Find a texture to use for this material online and save it to the Textures folder
   - Use a power of 2 size
     - 1024x1024, 512x512, etc
   - Make sure it's tileable

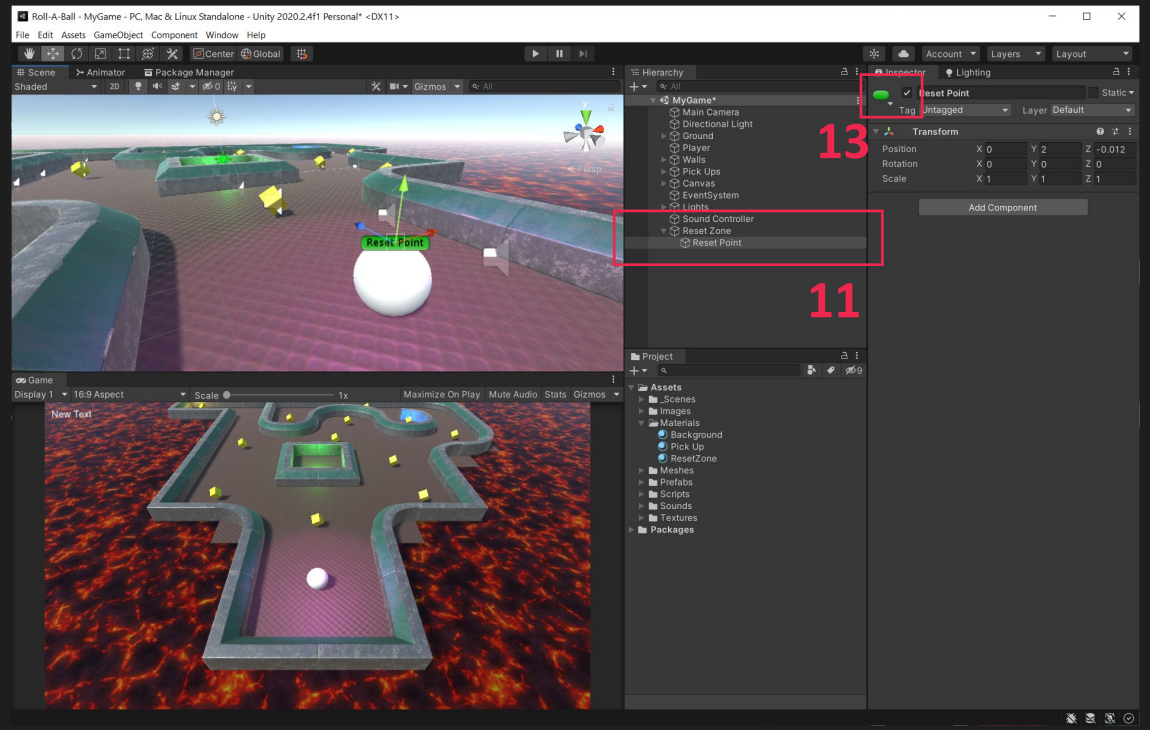
9. Drag the texture onto the Albedo slot on the Reset Zone material

10. Adjust the tiling to get it looking right



Setup the point to reset to

11. Create an empty GameObject as a child of the Reset Zone and call it Reset Point
12. Position it to where in the stage you will start from should you fall off the stage
13. Use the Icon next to the GameObjects name to make it easier to find in the scene



Modify our PlayerController script

14. Add the three variables near the top of the script

```
private Rigidbody rb;
private int count;
private int pickupCount;
GameObject resetPoint;
bool resetting = false;
Color originalColour;
```

15. Add the lines to the Start function

```
void Start()
{
    rb = GetComponent<Rigidbody>();
    count = 0;
    pickupCount = GameObject.FindGameObjectsWithTag("Pick Up").Length
                + GameObject.FindGameObjectsWithTag("Bowling Pin").Length;
    gameOverScreen.SetActive(false);
    SetCountText();
    winText.text = "";
    resetPoint = GameObject.Find("Reset Point");
    originalColour = GetComponent<Renderer>().material.color;
}
```

16. Add to the FixedUpdate function

```
void FixedUpdate()
{
    if (resetting)
        return;

    float moveHorizontal = Input.GetAxis("Horizontal");
    float moveVertical = Input.GetAxis("Vertical");
    Vector3 movement = new Vector3(moveHorizontal, 0.0f, moveVertical);
    rb.AddForce(movement * speed);
}
```

Modify our PlayerController script

17. Create a new function called OnCollisionEnter and fill it out like here
    - NOTE: You may already have the OnCollisionEnter function from another module. If so, just add the lines in the function to the existing one

```
private void OnCollisionEnter(Collision collision)
{
    if(collision.gameObject.CompareTag("Respawn"))
    {
        StartCoroutine(ResetPlayer());
    }
}
```

18. Create the Reset IEnumerator. This type of function allows us to pause execution of code until we want it to resume.

19. This function is quite lengthy but is so that the player smoothly moves back to the ResetPoint and not just instantly moves there.

```
public IEnumerator ResetPlayer()
{
    resetting = true;
    GetComponent<Renderer>().material.color = Color.black;
    rb.velocity = Vector3.zero;
    Vector3 startPos = transform.position;
    float resetSpeed = 2f;
    var i = 0.0f;
    var rate = 1.0f / resetSpeed;
    while (i < 1.0f)
    {
        i += Time.deltaTime * rate;
        transform.position = Vector3.Lerp(startPos, resetPoint.transform.position, i);
        yield return null;
    }
    GetComponent<Renderer>().material.color = originalColour;
    resetting = false;
}
```

DONE!!!

Remove a wall and have the player fall off the stage to reset their position.

You can set up other colliders within the stage that will also reset the player as long as they are tagged “Respawn”

3. Dynamic Number of pickups	0pts
Pre-requisite: None	
We will make the number of pickups in our game dynamic – That is allow us to add more or less pickups, and trigger different win conditions win based on what we have collected.	
Design: None	
Programming: Finding all objects with tag, modifying win condition	

Modify the PlayerController script

1. Declare an int called pickupCount that will hold the total number of pickups in our scene

1

```
private Rigidbody rb;
private int count;
private int pickupCount;    //The number of pickups in our scene
```

2. In the Start function, we use an inbuilt function to get the number of objects tagged “Pick Up” in the scene and assign that to the pickupCount

2

```
void Start()
{
    rb = GetComponent<Rigidbody>();
    count = 0;
    //Find all the objects with the tag Pick Up and set the count to that number
    pickupCount = GameObject.FindGameObjectsWithTag("Pick Up").Length;
    gameOverScreen.SetActive(false);    //Turn off our game over screen at start of game
    SetCountText();
```

3. We change the check condition from a hard coded number (12) to the pickupCount variable.
This way, if we add or remove Pick Ups, the value automatically adjusts to call our game over screen

3

```
void SetCountText()
{
    countText.text = "Count: " + count.ToString();
    if(count >= pickupCount)
    {
        gameOverScreen.SetActive(true); //Turns on our Game Over Screen
        winText.text = "You Win!";
    }
}
```

4. **And DONE!!!!**
That was almost too easy…

NOTE: There is other code between these sections that has been omitted to save space

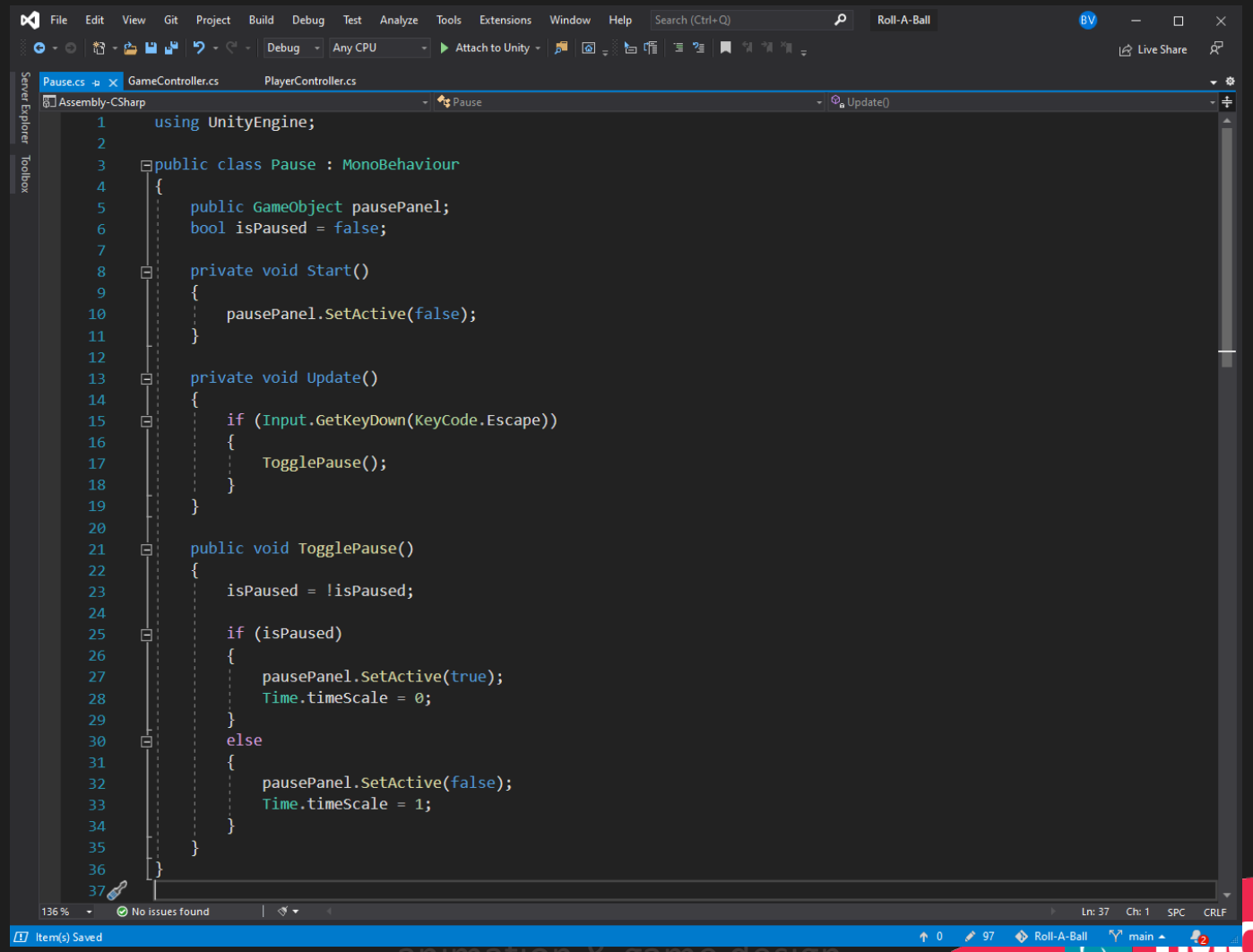
4. Pause & Restart Screen	0pts
Pre-requisite: Start and Game Over Screen	
Create a pause screen so we can exit and restart a level from within the level	
Design: UI Panels	
Programming: UI Buttons, Scene Management	

Create a Pause script

1. Create a script called Pause and write it out EXACTLY as follows

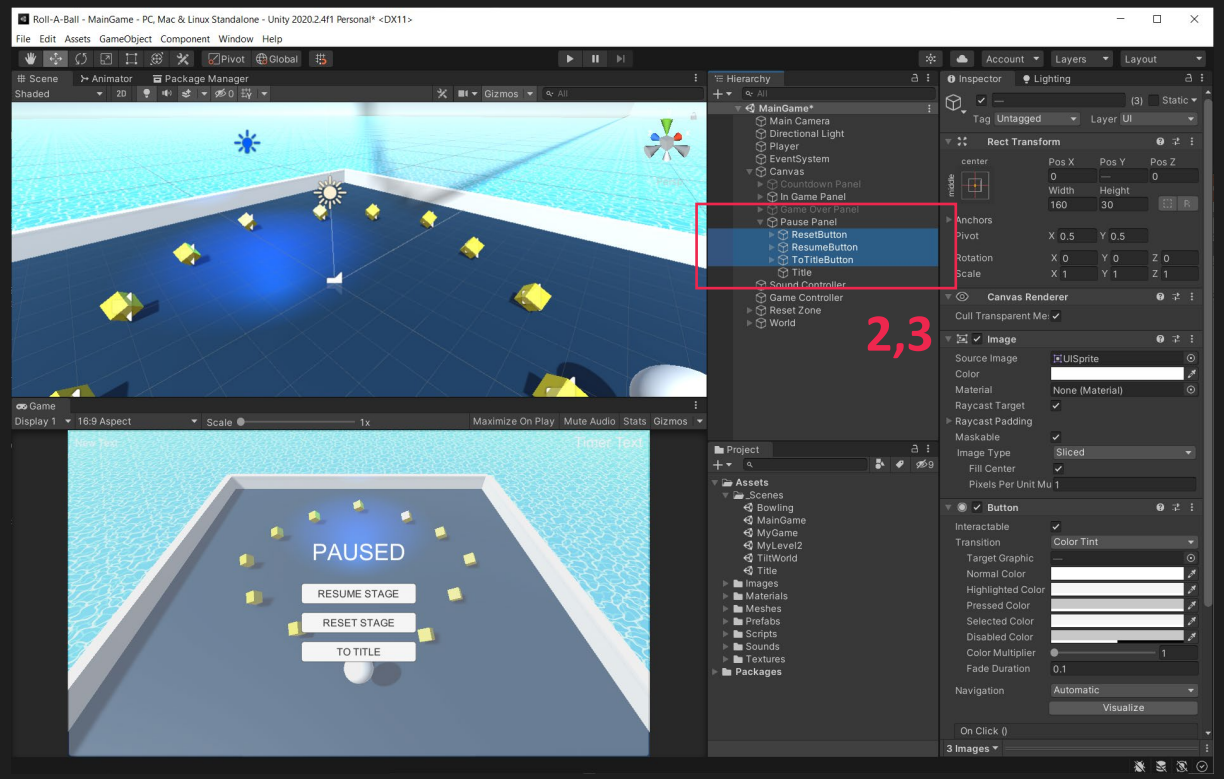
*Amendment
In the PlayerController script, add the following as the first line within the Start function:

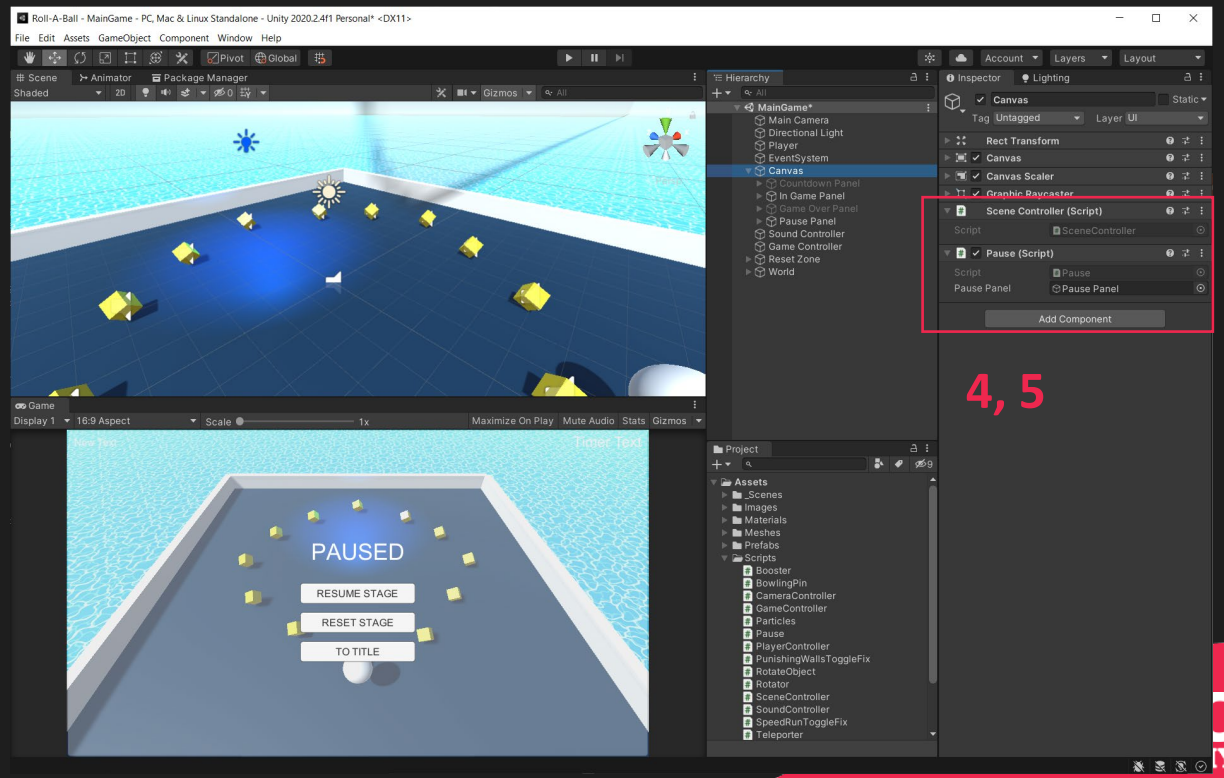
Time.timeScale = 1;



Create the buttons

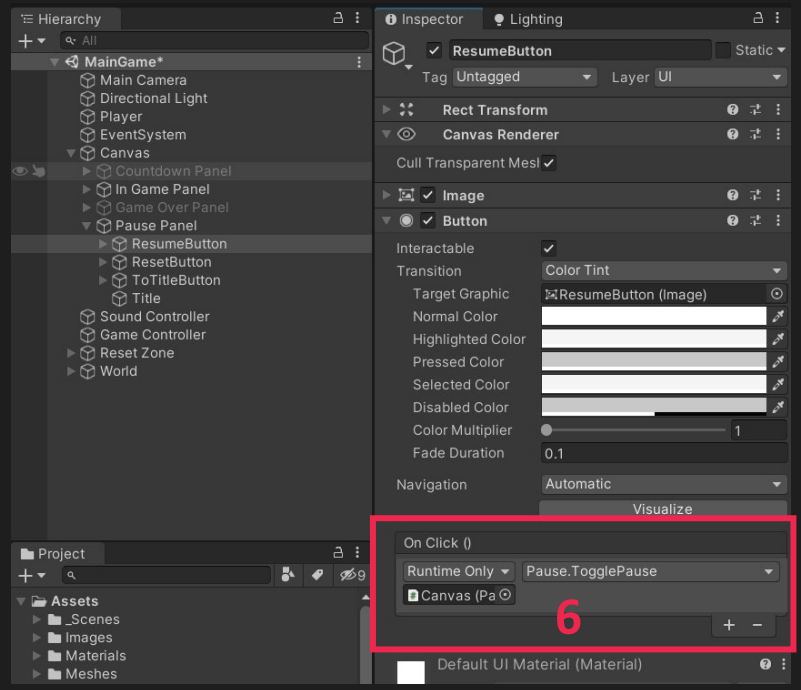
2. Create a UI Panel in the Canvas called Pause Panel
3. Add 3 Buttons to this panel and a title PAUSED
   - Resume Stage
   - Reset Stage
   - To Title
4. Add the Pause script to the Canvas Game Object.
5. You should already have the SceneController script from Module 1 also on the Canvas. If not, complete that step first!



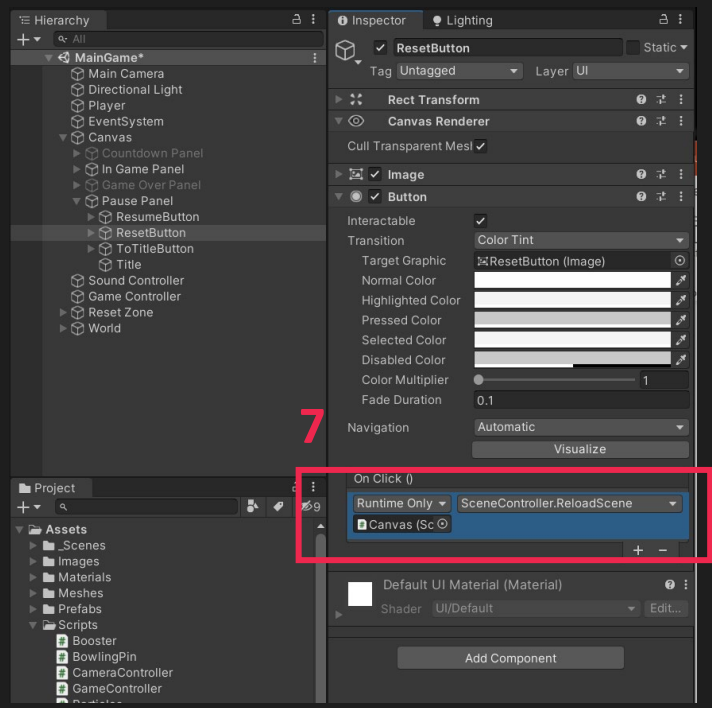


Hook up the buttons

6. In the Resume Buttons On Click,
   - click the +
   - drag the canvas game object to the slot
   - Choose Pause > TogglePause() from the dropdown



7. In the Reset Buttons On Click,
   - click the +
   - drag the canvas game object to the slot
   - Choose SceneController > ReloadScene() from the dropdown



8. In the To Title Buttons On Click,
   - click the +
   - drag the canvas game object to the slot
   - Choose SceneController > ToTitleScene() from the dropdown