Immersive Environments Tutorial

Goals:

- Rendering items

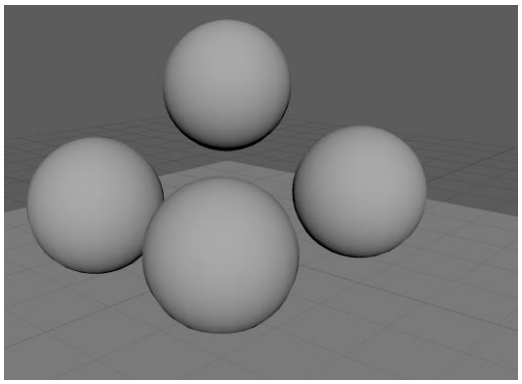
Load up Maya

Build Object: Rendering

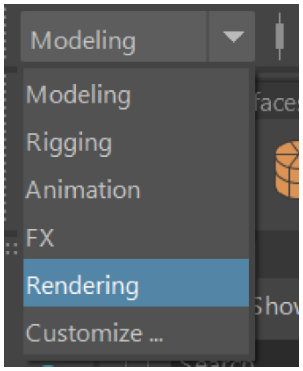
Aim: Create elements and render to external file formats.

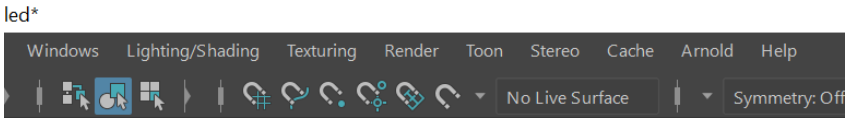
Rendering is the method of extracting the objects in the scene to an external file, such as a jpg/png.

Create a simple scene so we can test rendering.

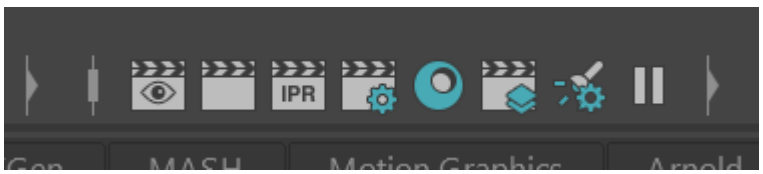


From here, go to the drop down section and select rendering, this will modify the menu system





There is also a quick rendering set of icons in the UI.



Let's look at these icons

This is the open render view. The render view is where you will be able to see all of the renders and have the ability to determine the render engine used.

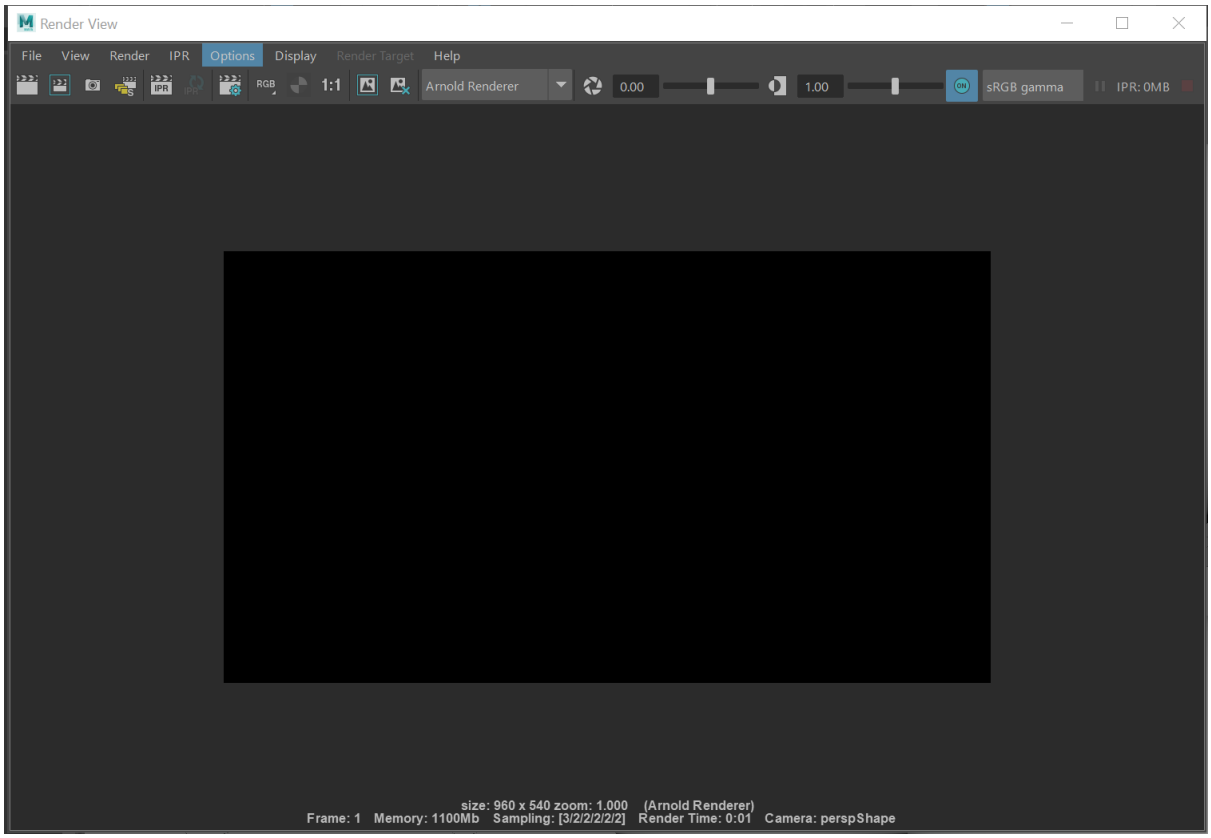
This is the render current view. The view is what you see in the scene. So, if you are in perspective view, that is what you'll see, side view, you'll see the side view and so forth. When swapping between render engines, you will have to re-render the scene.

This renders the current scene using Arnold. Arnold is the photo realistic render. It uses the most processing power as it calculates the light off an object through three differing aspects of where light can be reflected. Arnold elements will be seen through this render.

This is the render settings, i.e. where you are putting the file, what file type you are creating.

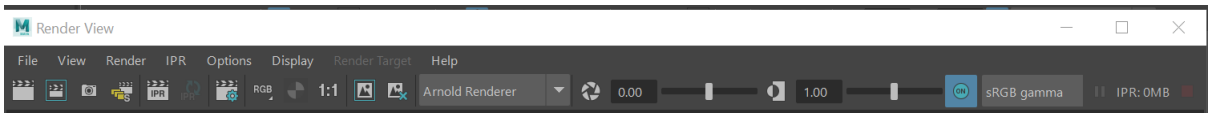
This is the hypershade system. This area lists all of the materials in the scene, and allows you to preview the materials on a set item quickly.

To start with click on the Render current view icon. This opens up the render view.

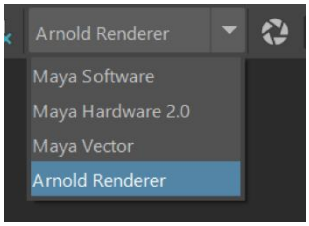


The default render is Arnold. As you can see, the scene is pure black, this is because we haven't applied any Arnold textures.

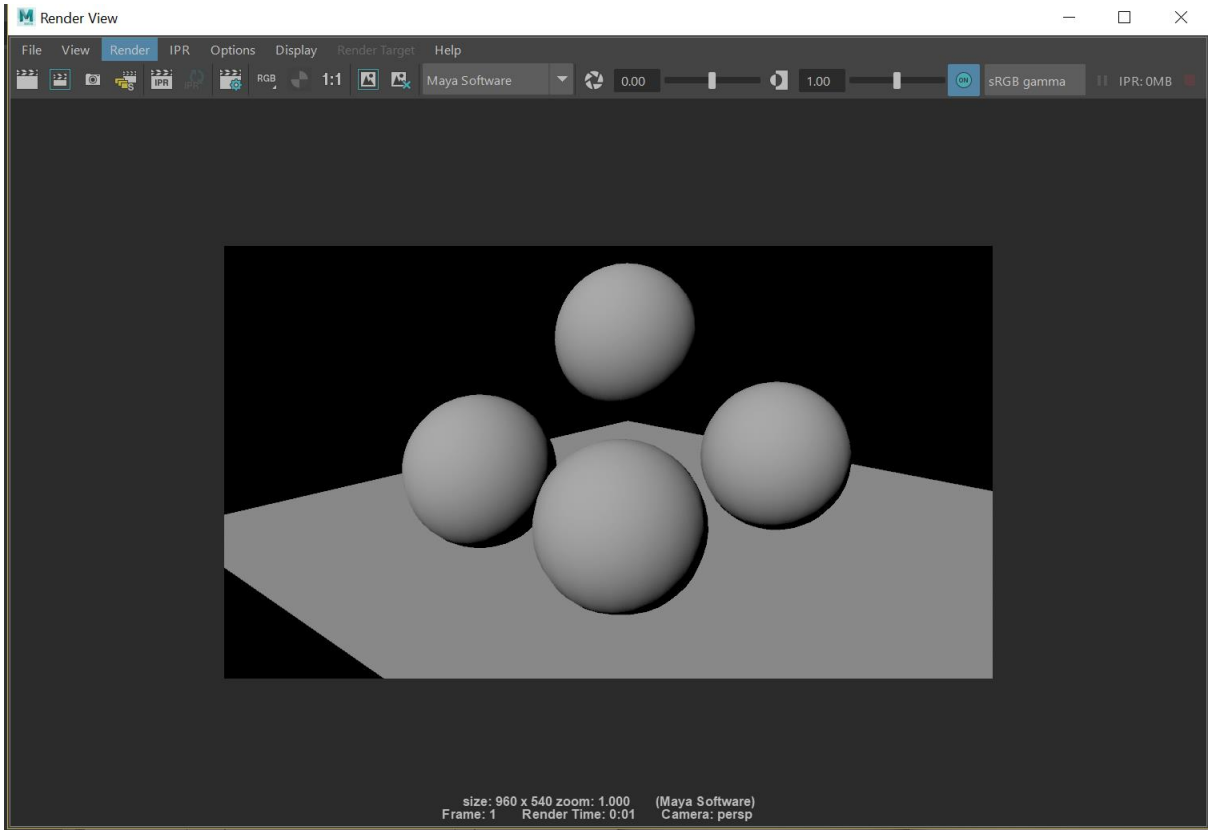
The menu system of the render window allows you to change a lot of elements



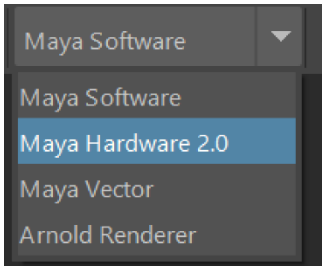
From the drop down, we will go through the render engines.



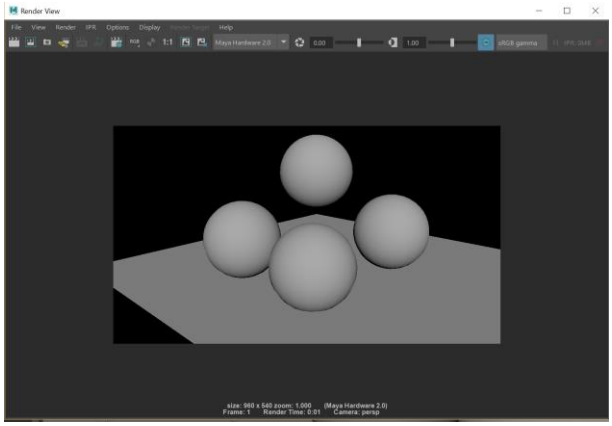
Change the render engine to Maya Software and click on the render current scene icon.



As you can see, because it is a simple method of implementing rendering, the light bounces off the surface, our base textures resolve and we can see the scene. Next swap to Maya Hardware.

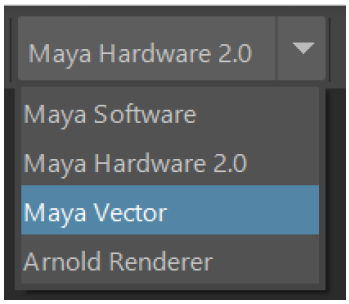


As before, we've changed engine, now click to re-render the scene.

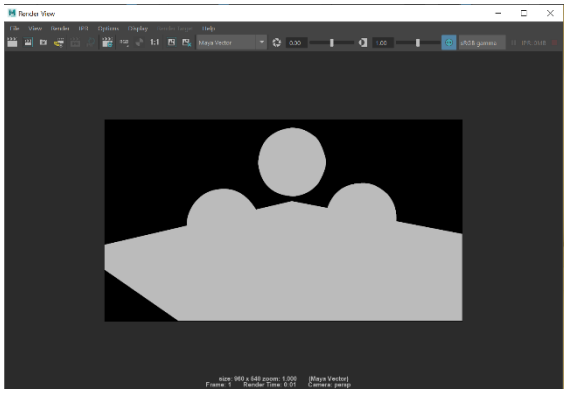


As you can see, the image rendered very quickly. This is a great way of pushing the rending to a graphics card for intense rendering scenes.

Finally, we look at Maya Vector.



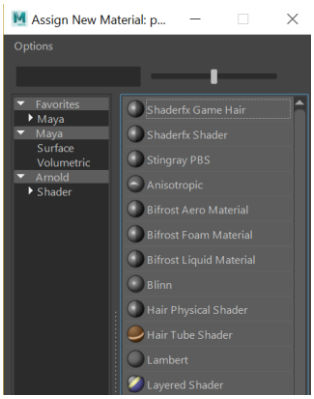
Remember to render the scene.



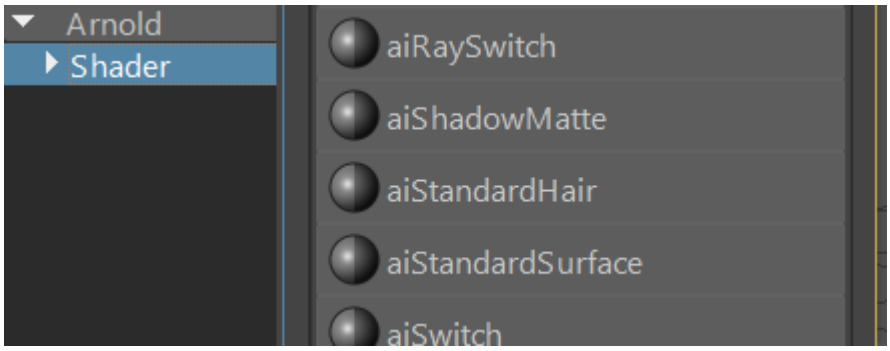
Notice, that the details have left the rendered objects and we are left with the external shape of everything.

Now, let's add some materials to our scene. Let's use the menu system for this, which is lighting/Shading -> Assign new material. You could always right click on the object and select assign new material. Do this on the front most sphere.

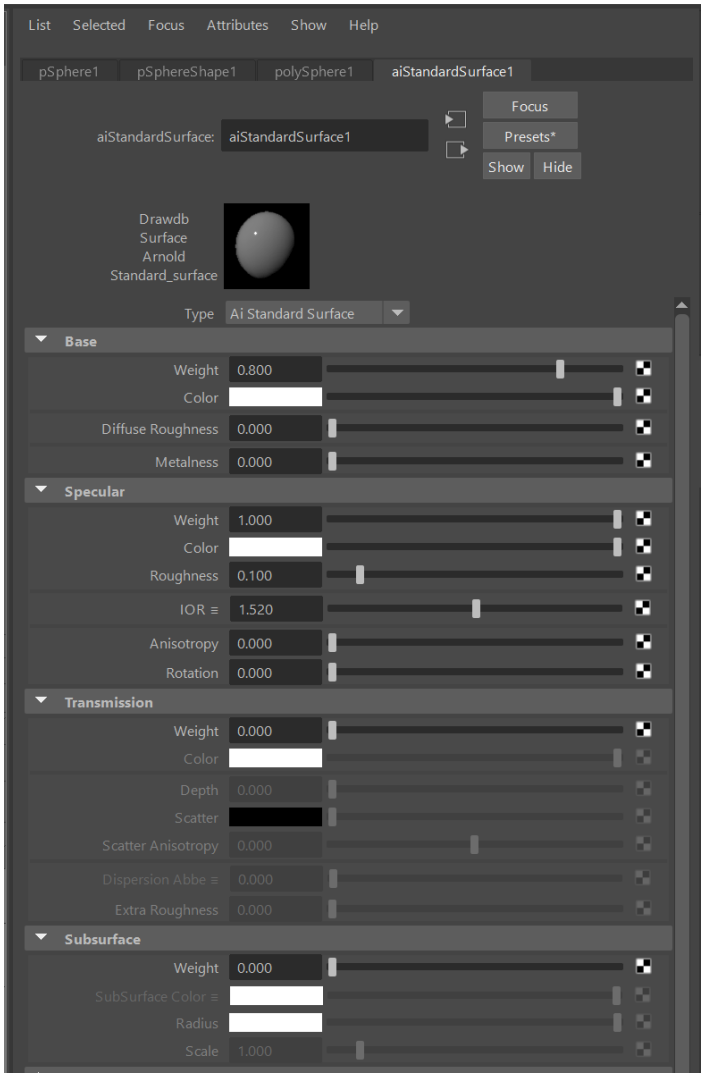
This gives the following menu system



From the side menu part select Arnold and shader, then from the right menu choose aiStandardSurface.

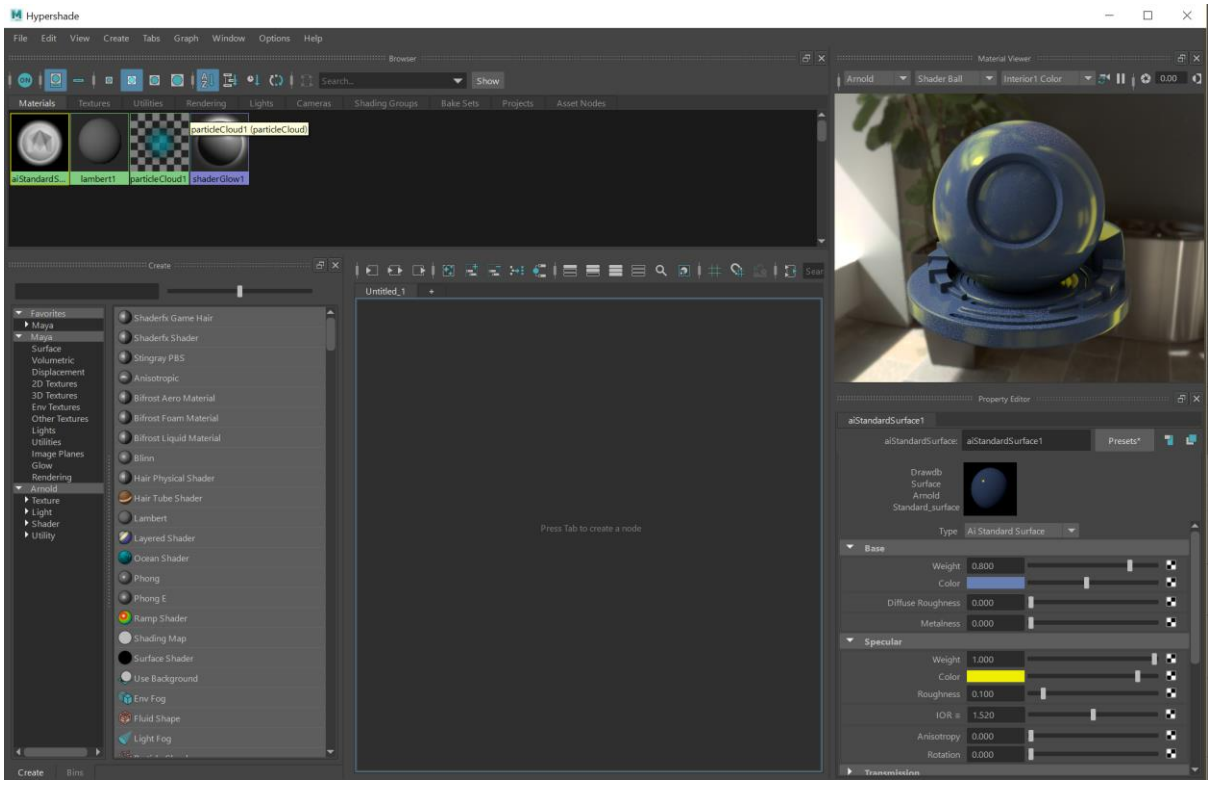


From the attribute editor there are 4 elements to the shader, they are Base, Specular, Transmission and Subsurface.

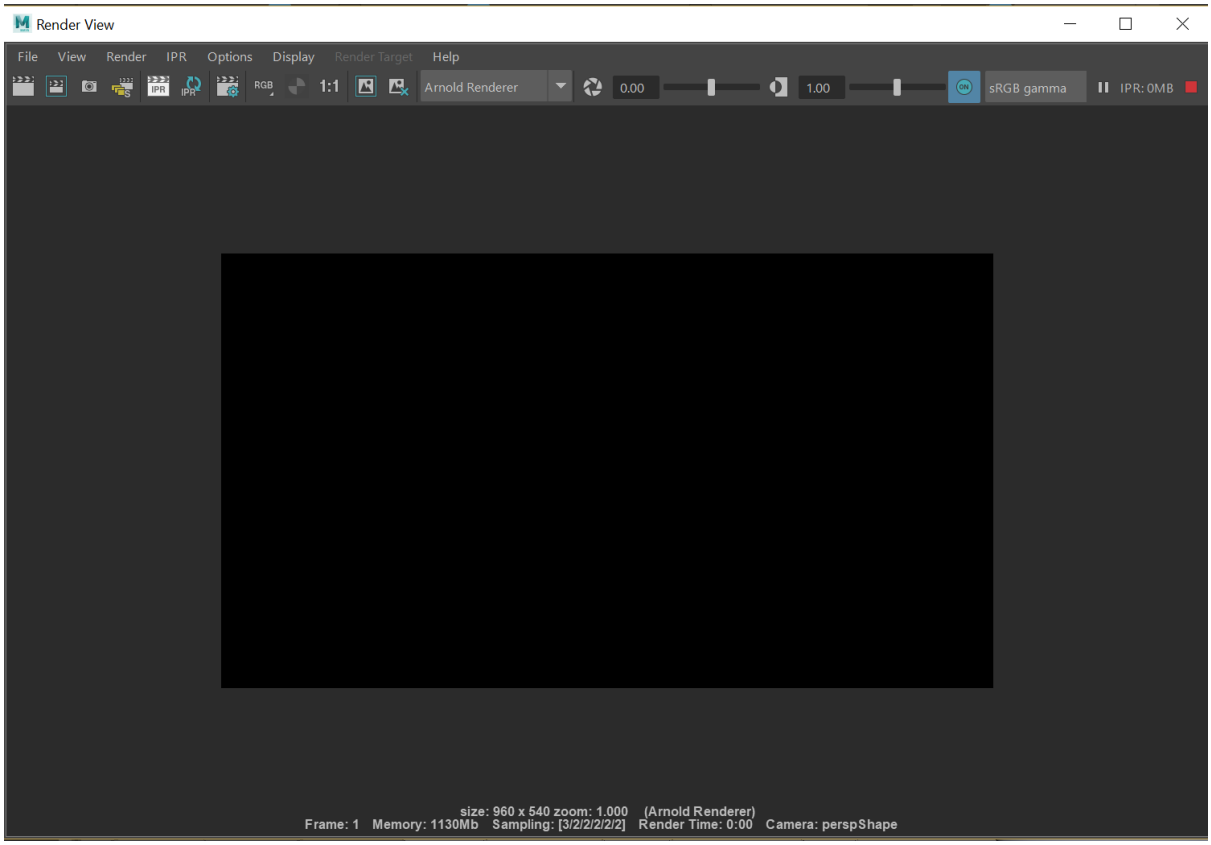


From here, play with the base and specular colours to get a feel for how these elements will be rendered.

Instead of rendering, go into the hypershade and see what the material selection would look like.

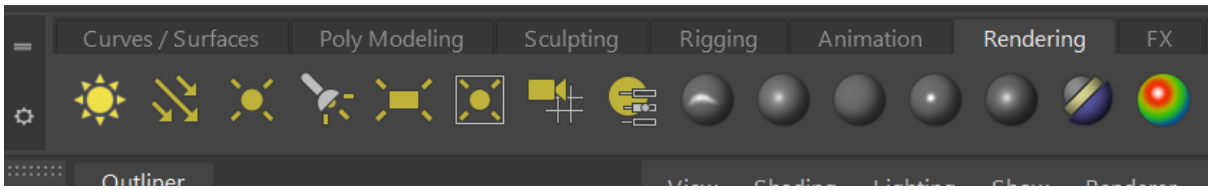


From here, you can manipulate the elements of the material and view it's look type. Now, go back to the perspective view and render the current scene using the IPR icon.

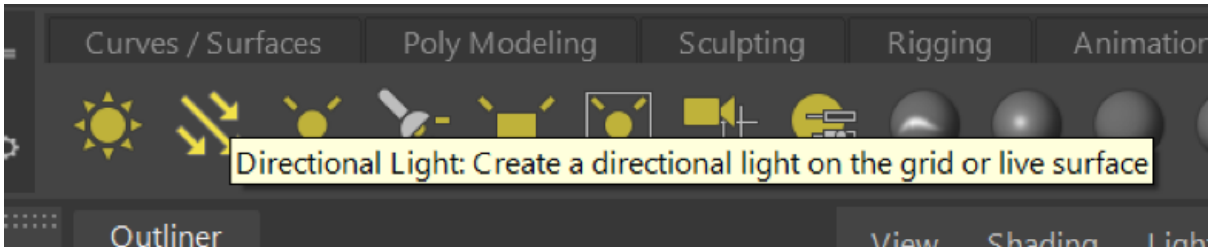


The scene renders blank, this is because we are starting to use Arnold textures we need to think about the lights that are occurring. Arnold appears to not like the default light system, so we will put in our own directional light.

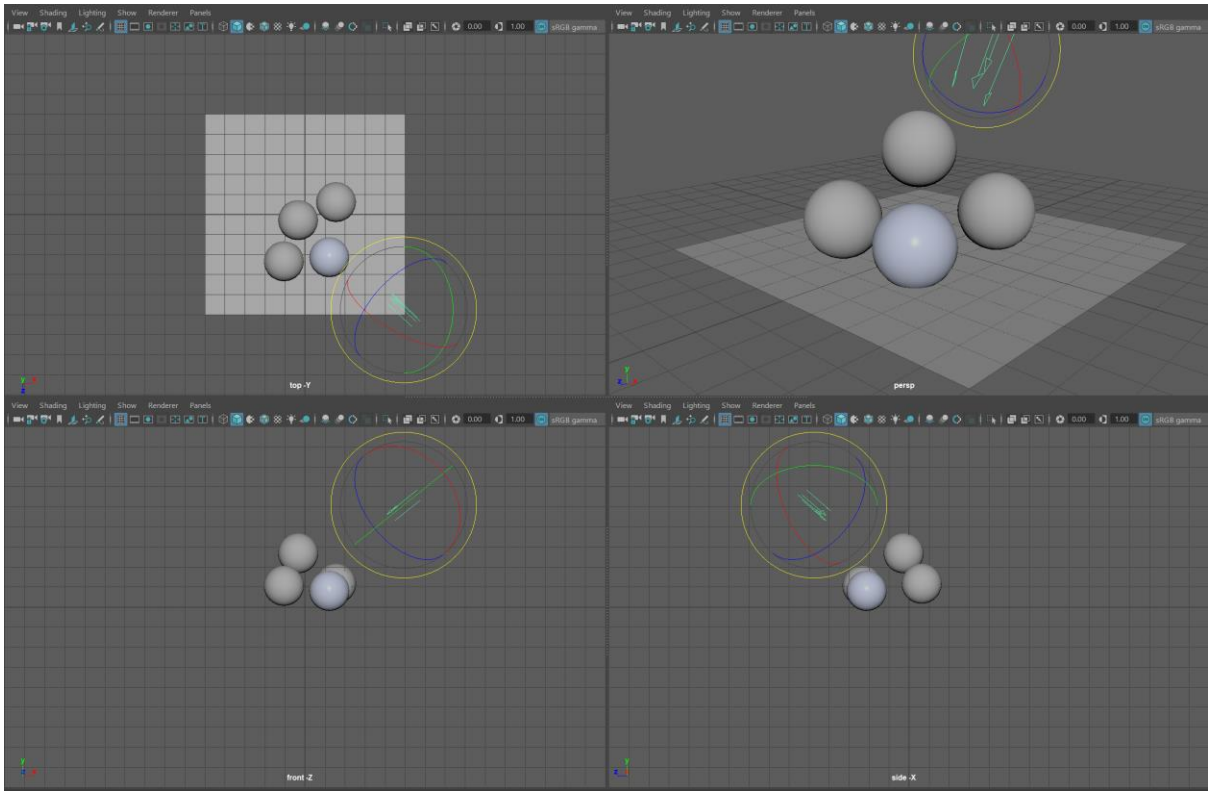
Click on the rendering tab



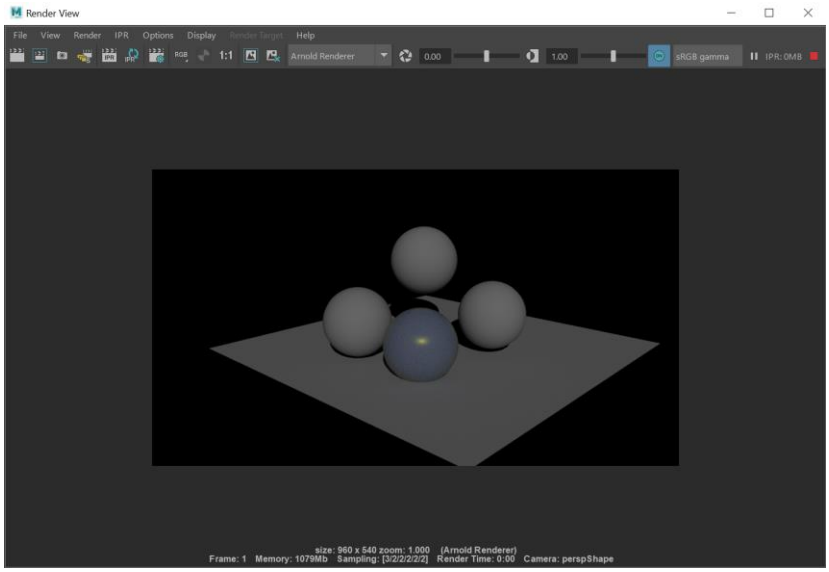
Then click on Directional Light.



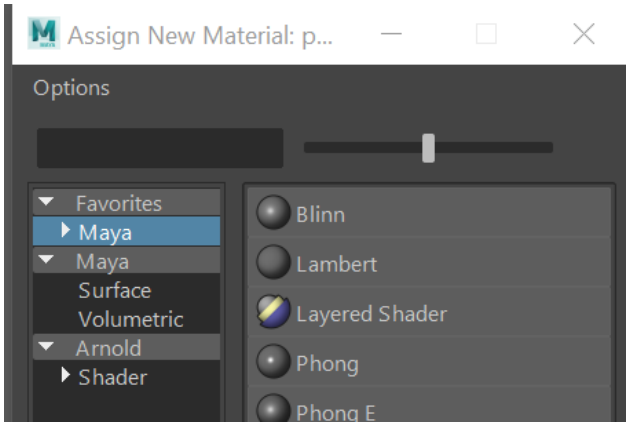
Treat this light like a standard object and move it so that it faces the spheres.

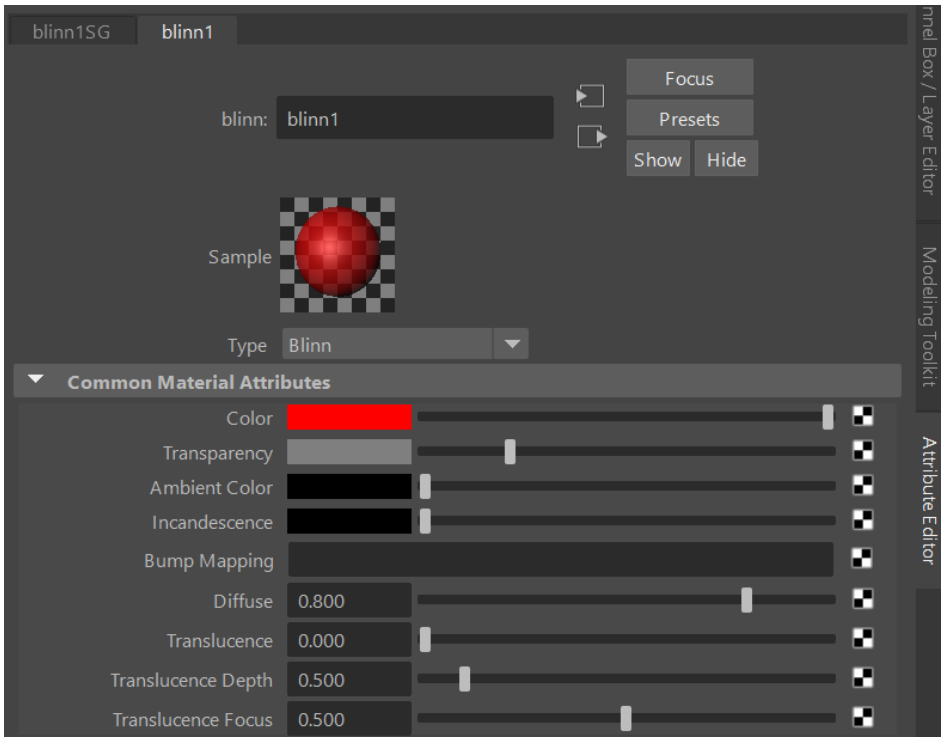


Once positioned, use the IPR icon again and see the rendered output.

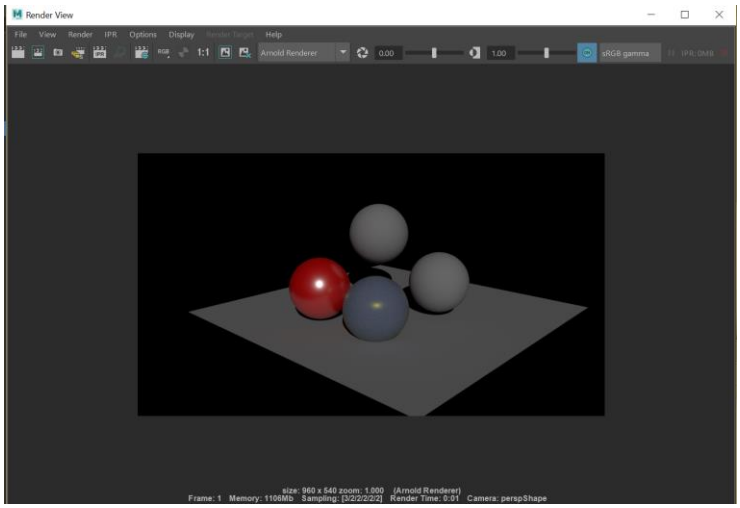


Select another ball and apply a blinn material.





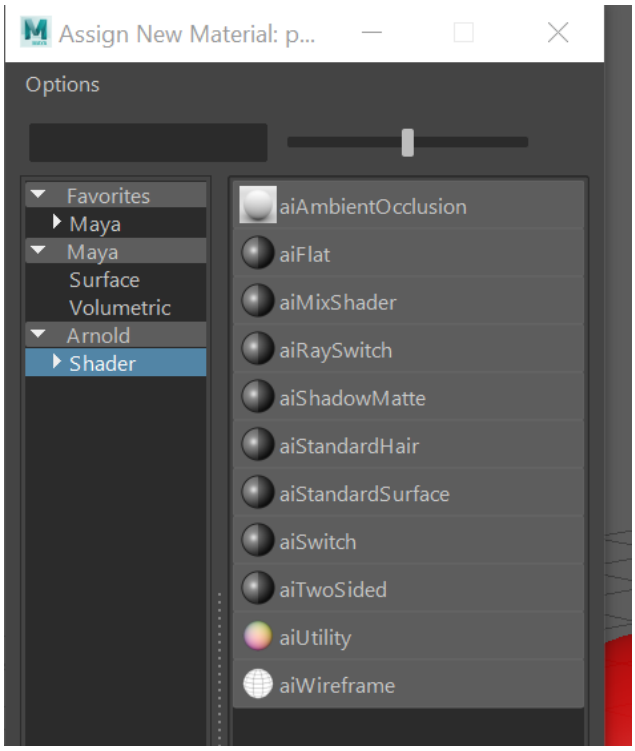
Render the scene again



Notice that IPR rendering handles blinn objects quite well, if you swap to other render engines and test, you will see that the elements are not rendered so well.

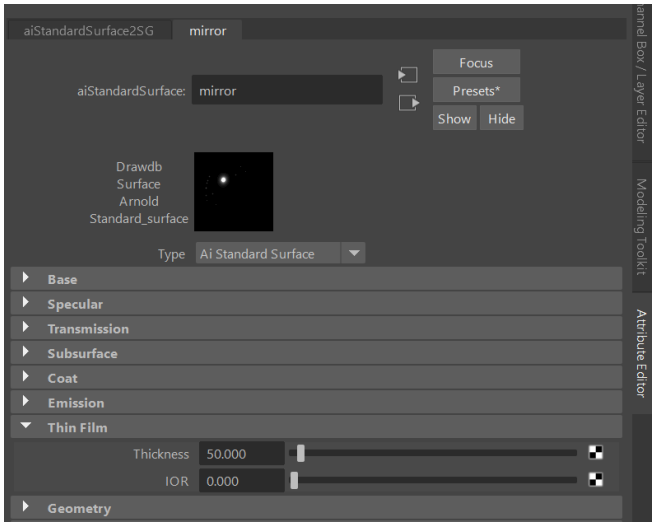
Next, we will make the middle ball a mirror ball.

Right click the floating ball and assign a new material. Select aiStandardSurface

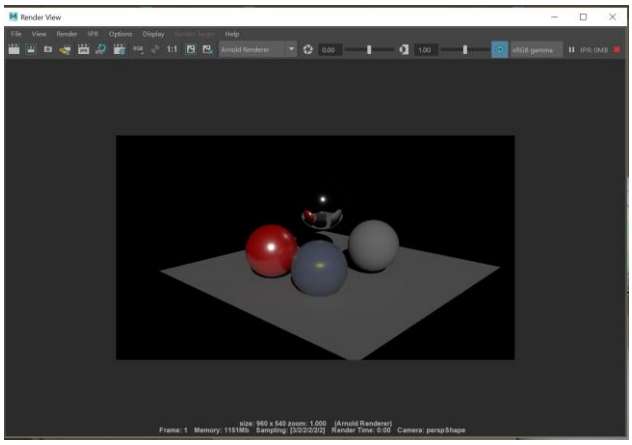


Name the material mirror, then scroll down through the attributes, find the Thin Film section, change the thickness to 50 and drop its IOR to 0. The IOR is the refractive index, a website that let's you see more about this is:

https://support.solidangle.com/display/A5AFMUG/Thin+Film

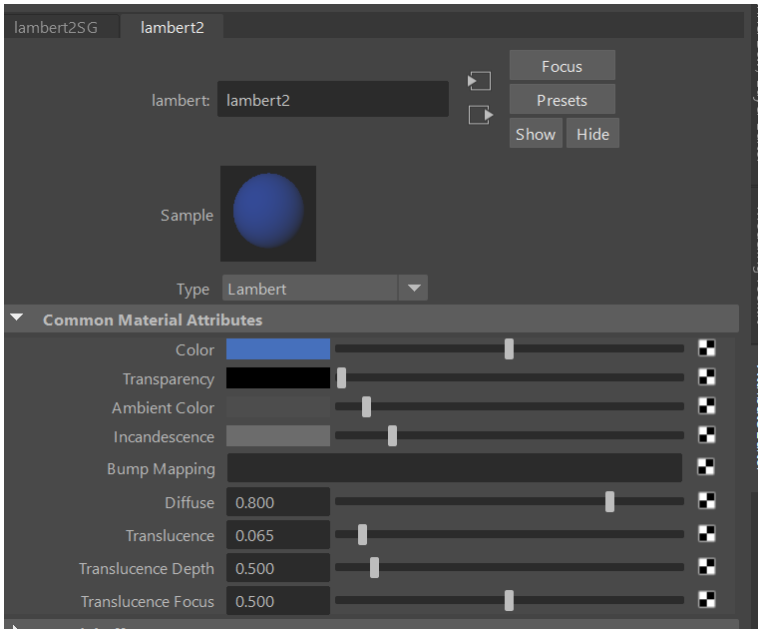


Once done render the image

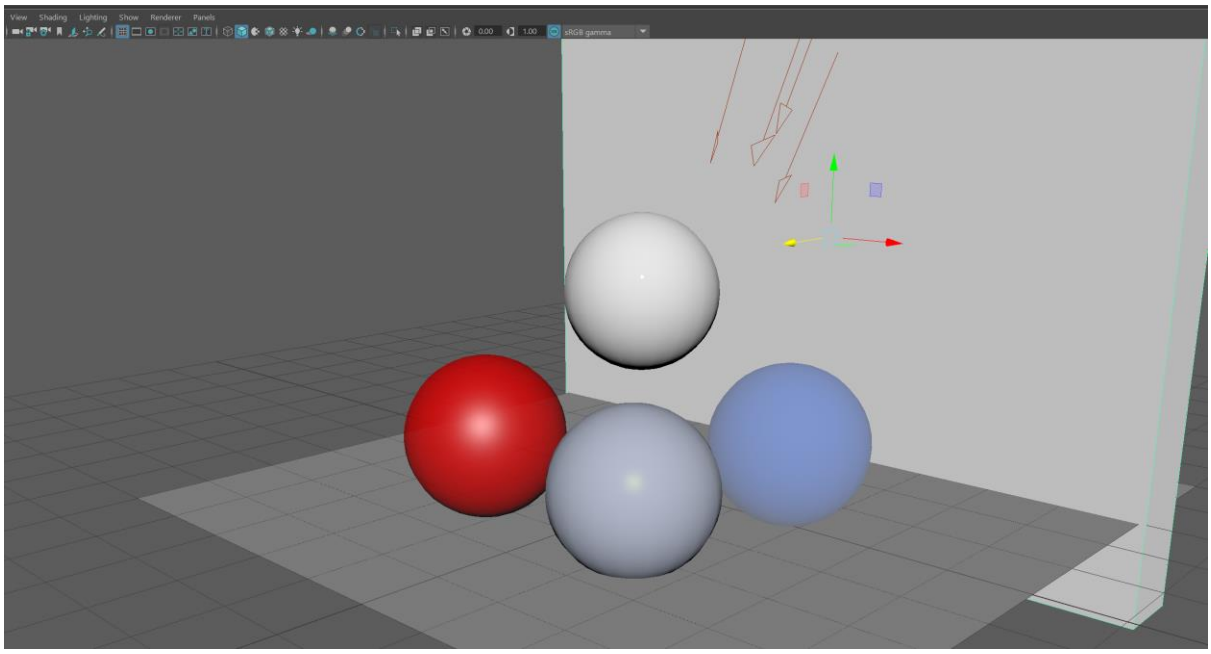


You should have a sphere reflecting the other spheres.

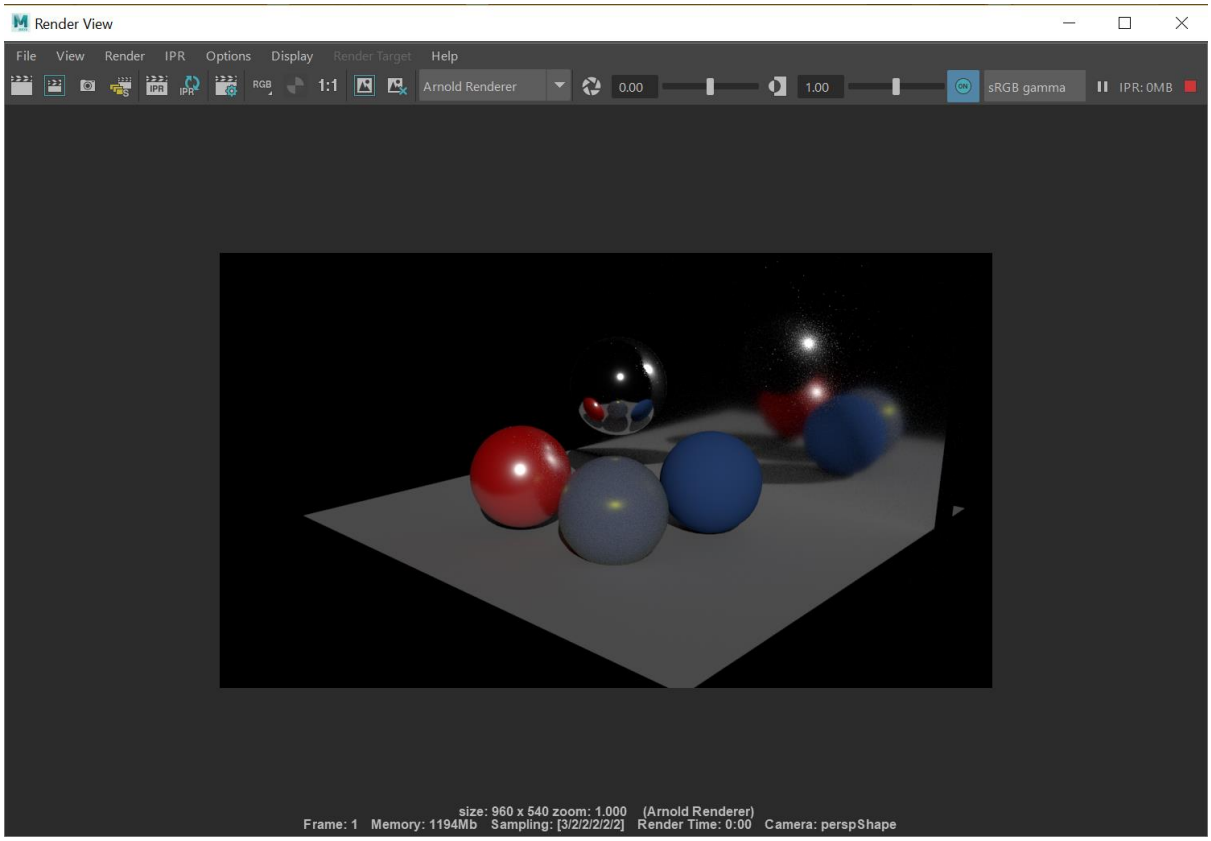
On the final sphere, place any material you want. In this case, I've just applied a lambert material with a blue colour



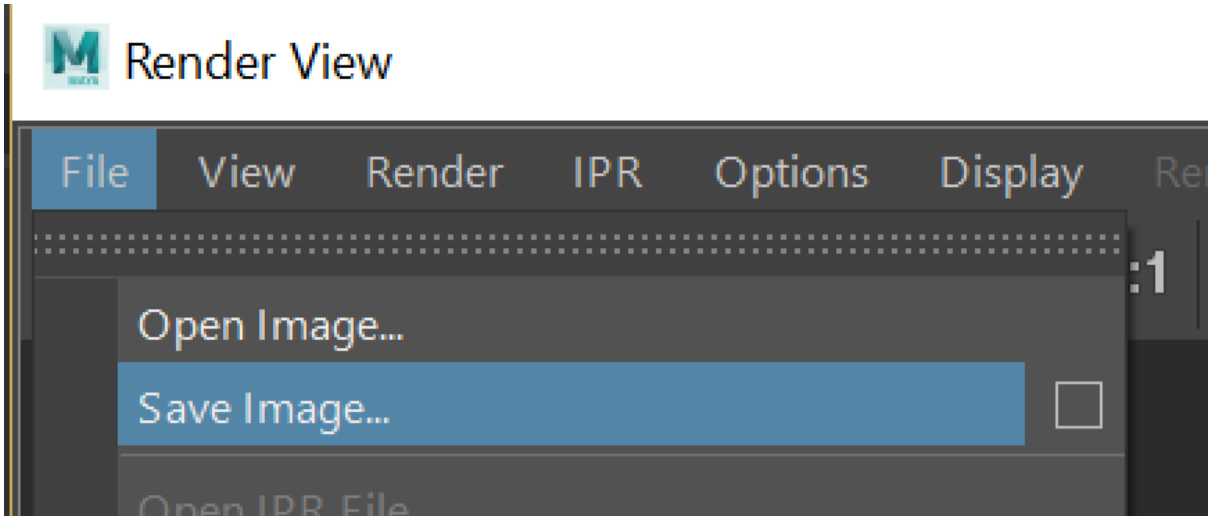
Next add a cube wall behind the new sphere on the plane and apply the mirror material. You should end up with the following.



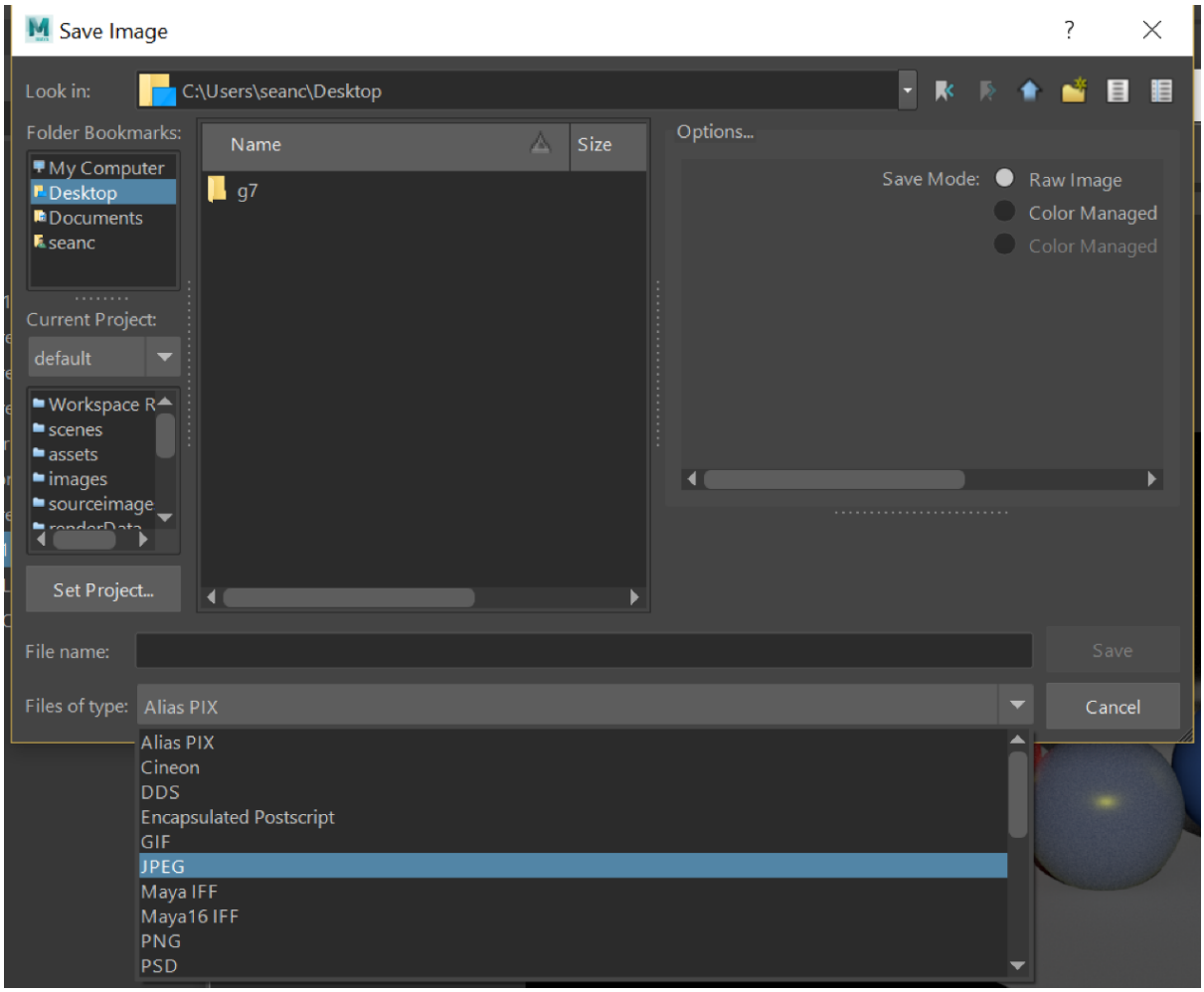
Which renders out like this:



To save the image, in the render view go to Menu file->Save image.



Select where you want it and what file type.

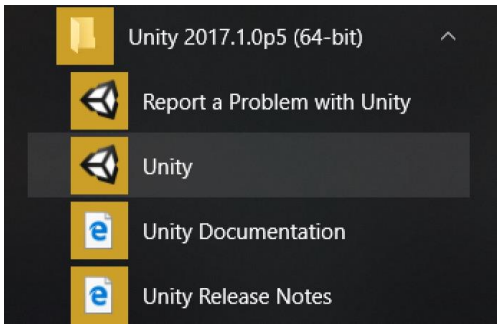


Unity

Goals:

- Scenes and collision triggers

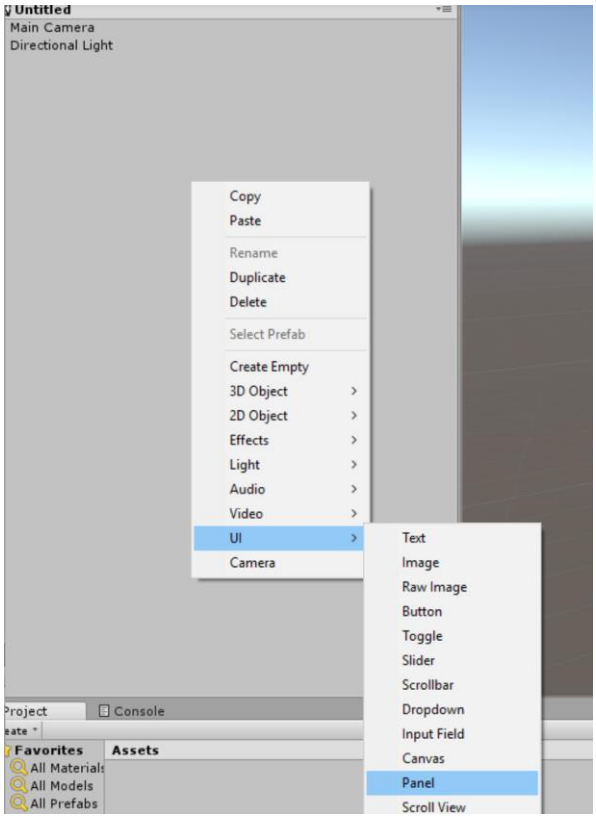
Load up unity



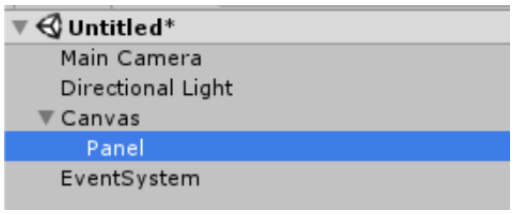
Build Object: Scene Swapping and Menu Creation

Aim: Create multiple scenes and swap between them, i.e. menu systems

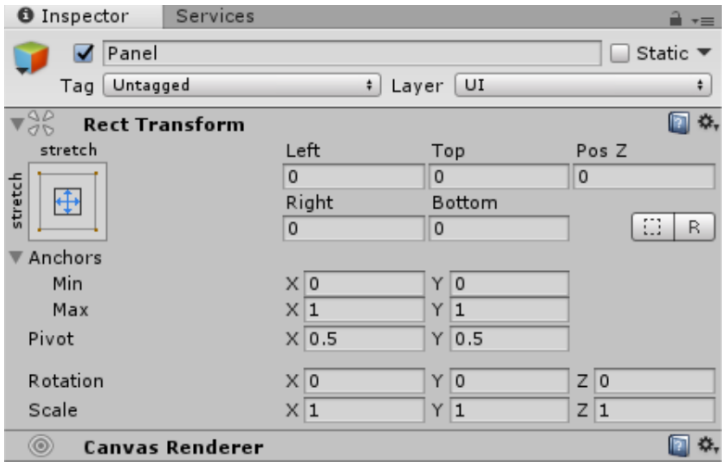
To start with right click in the hierarchy and create a Panel, this is done via the UI element.



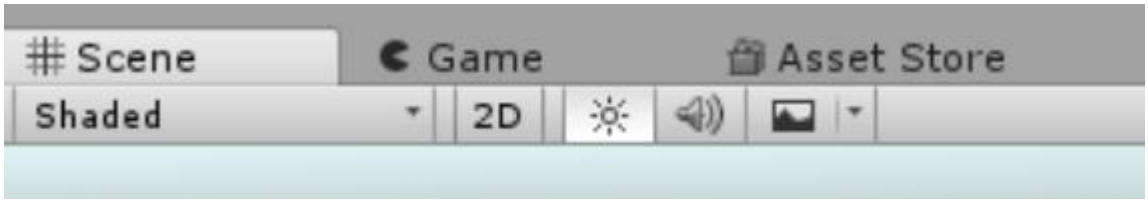
When the panel is created you will see that it adds an Event System and Canvas. The Canvas is designed to hold all elements of the User Interface.



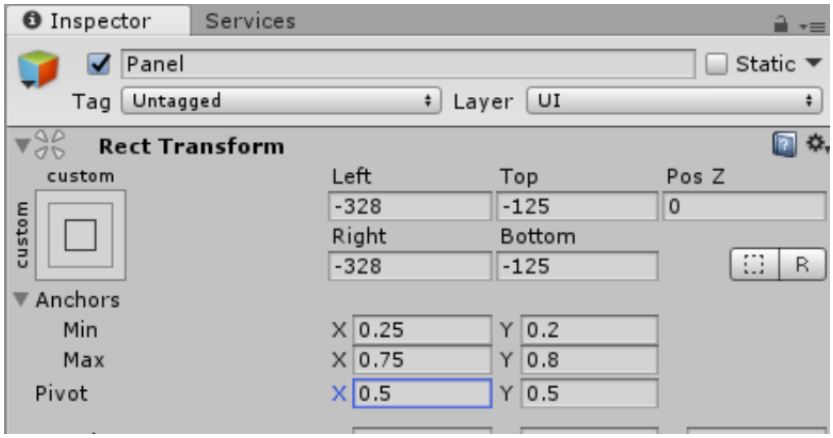
On the Inspector, the Panel has a lot of elements such as Anchors. Anchors a re a way of holding the elements inside the panel, the easiest way to set up the screen is to set the anchors and then, change the Left, Top, Right and bottom values.



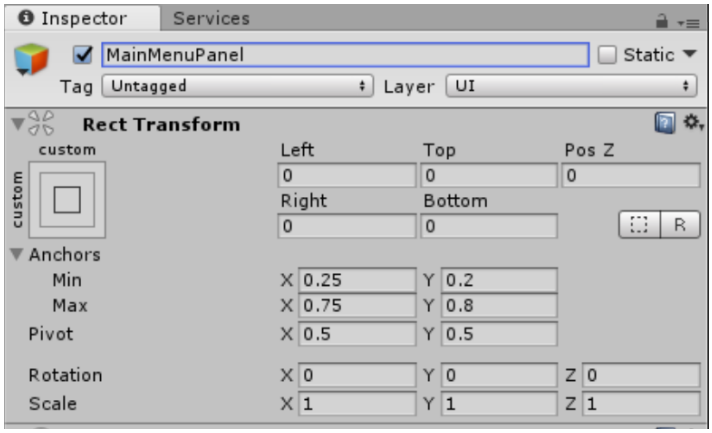
Change from 3D to 2D to be able to see the scene easier. Click on 2D.



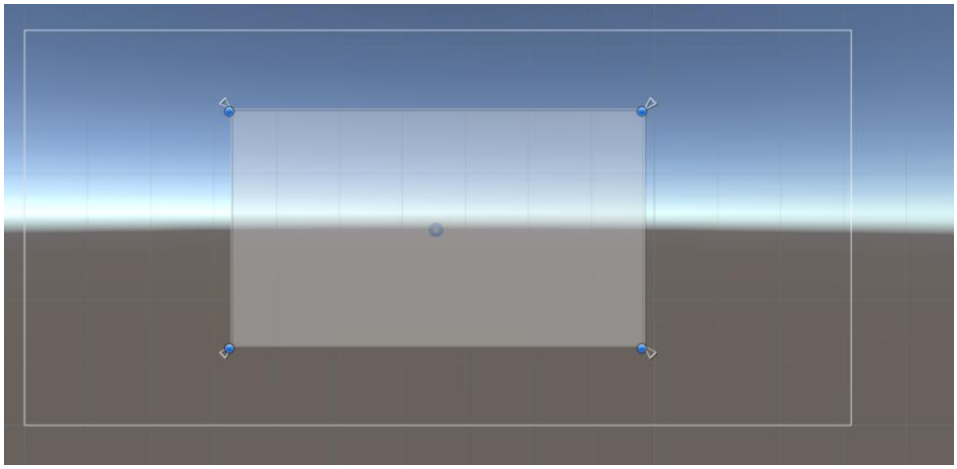
Next change, the Anchors min max values respectively to X 0.25 & 0.75 and Y's values to 0.2 & 0.8. Once that is done, you should see the following.



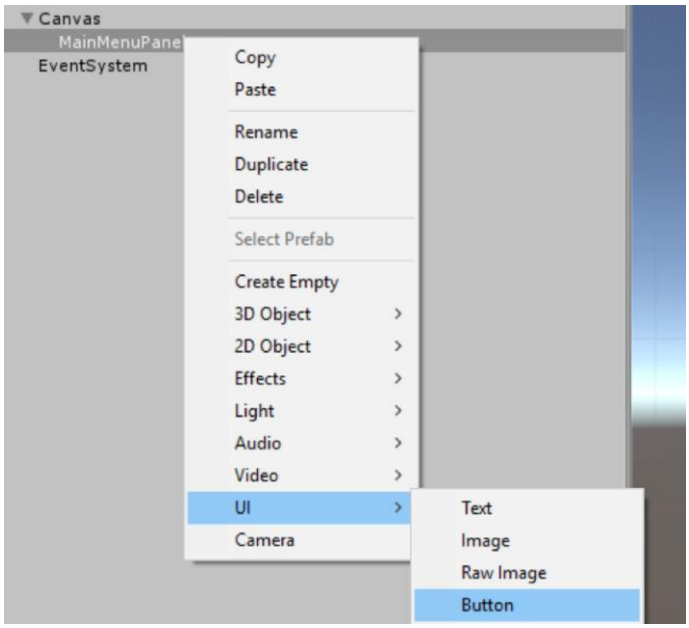
From here zero out the Left, Top, Right and Bottom values. Also, change the name of the panel to MainMenuPanel. Then scroll to examine the scene.



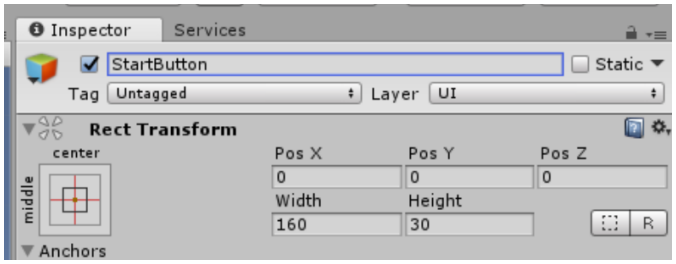
These values centre the panel into the screen, as can be viewed below. The white box is the viewing area.



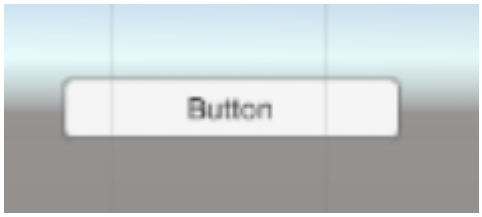
On our Menu system we will have three buttons, Start Game, Help and Quit. To start with, right click on the hierarchy panel and create a button as a child to the MainMenu Panel.



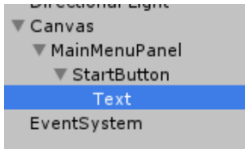
Once created, give this first button a name such as StartButton, in the Inspector panel.



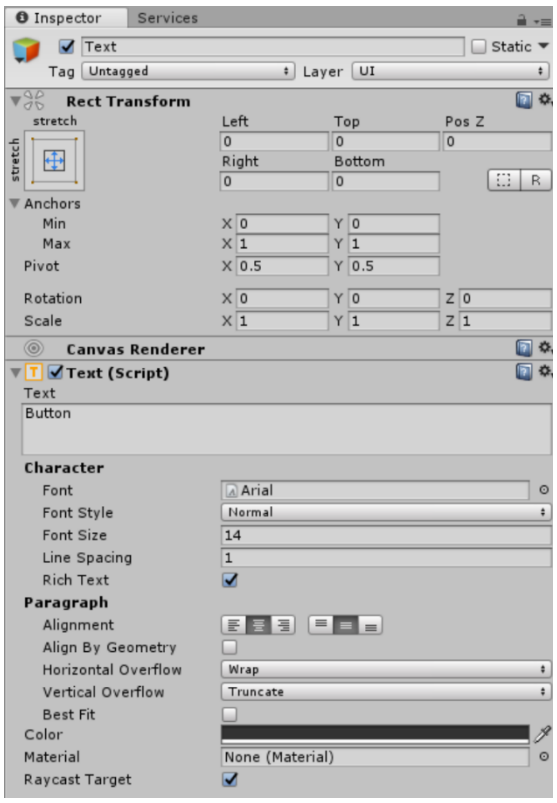
Notice in the Scene view that the button has the text of button on it.

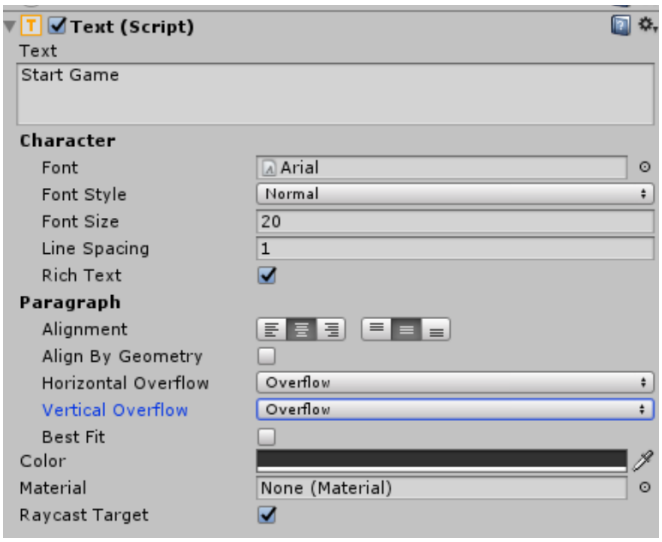


To change this, go through the hierarchy panel to get to the text element of the button.



From the inspector panel, you can change the text, font style, font size and how text acts. In this case change the text from button to Start Game, increase the font to 20 and change both vertical and horizontal overflows to overflow.



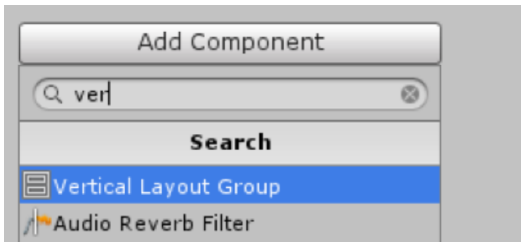


When you look at the scene view, the button is centred on the layout and the text fills the scene.

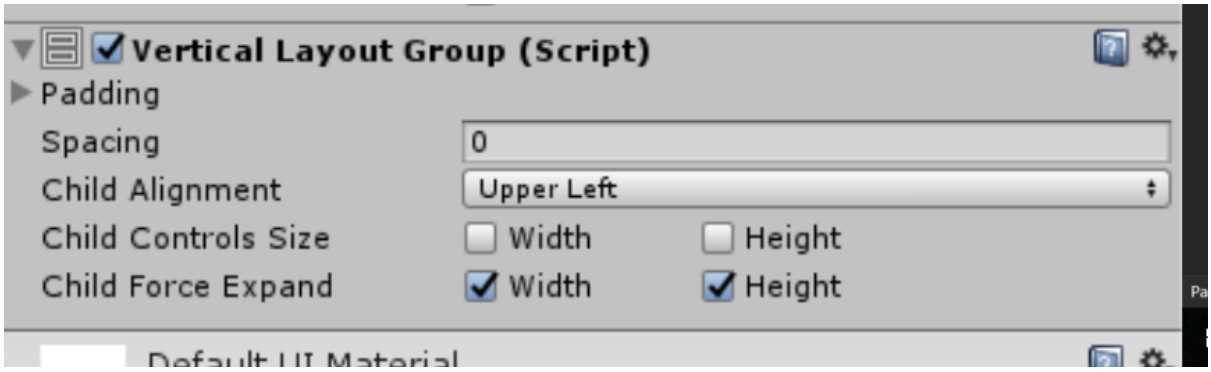
Next, we want to create the other two buttons, we could create a new button and go through the same steps, but then we would be dealing with the alignment of the buttons. The best thing to do is to add a vertical layout group to the

MainMenuPanel. Click on MainMenuPanel on the hierarchy and then add the new component to the inspector for it.

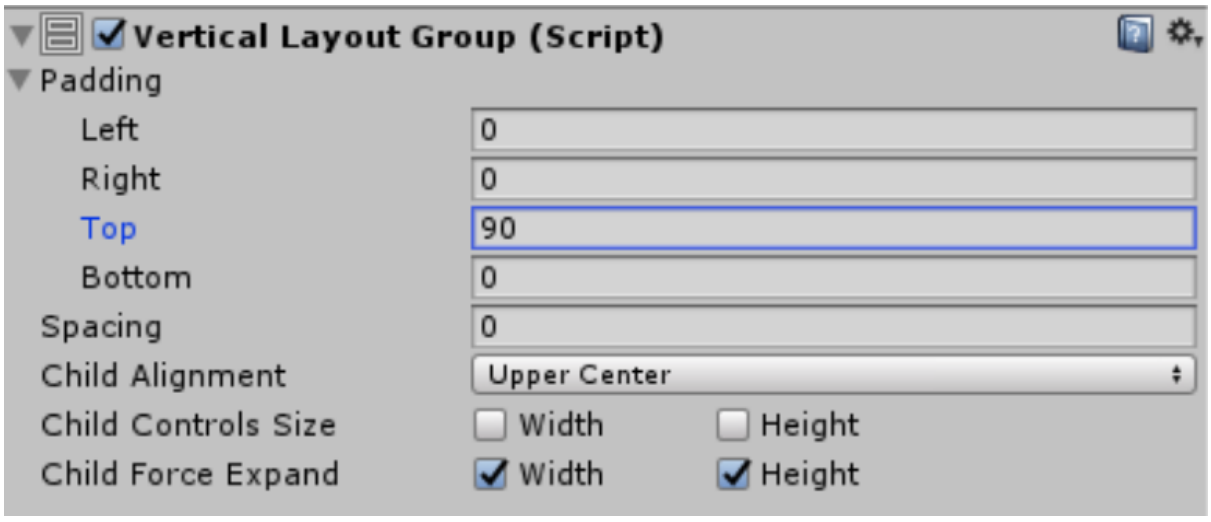
To locate use the search capability.



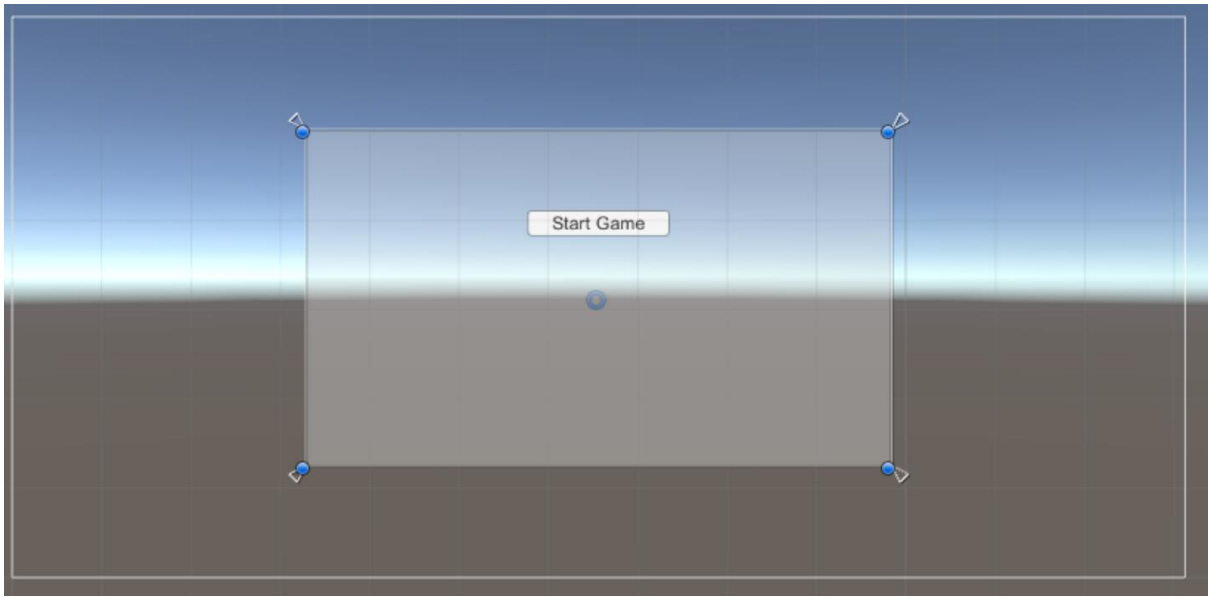
This automatically puts the button in the top left corner of the panel, which is not what we want to do, as such, we can make adjustments to the Vertical Layout Group to position it.



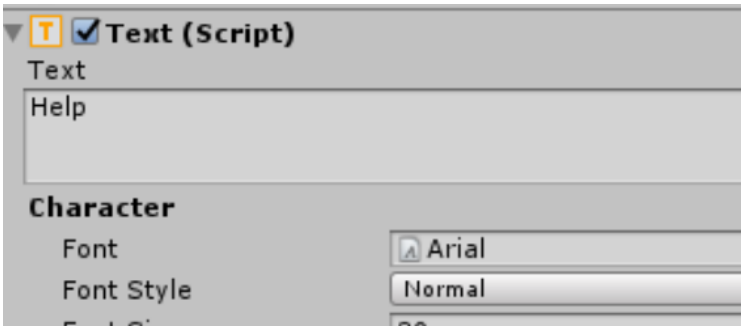
The changes are from the drop down menu, change it to upper center and add a top padding of 90.



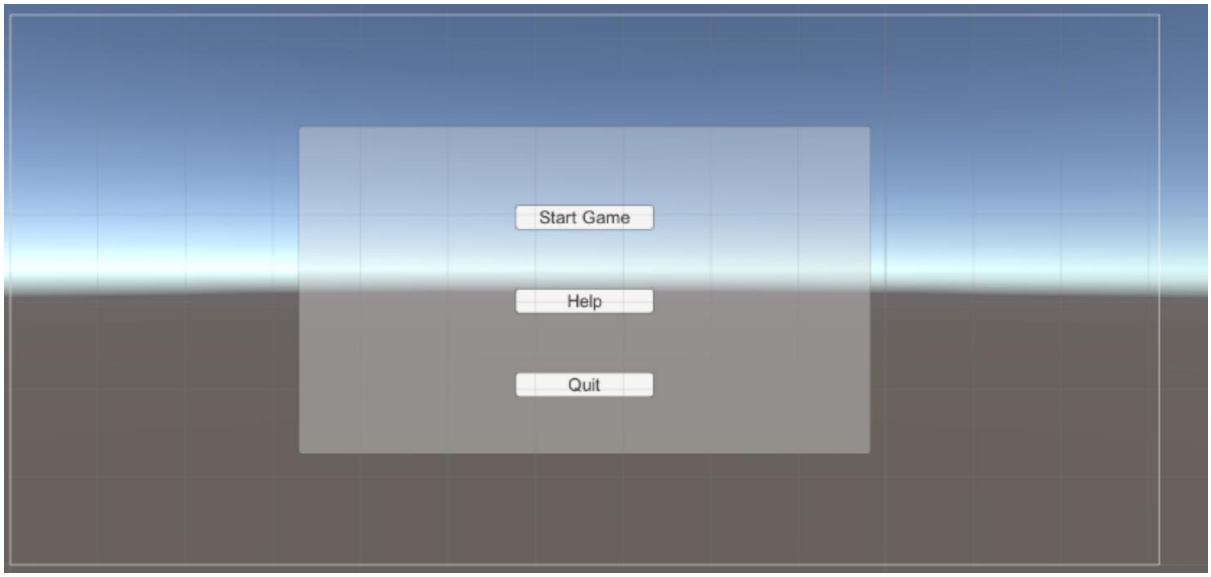
This produces the following



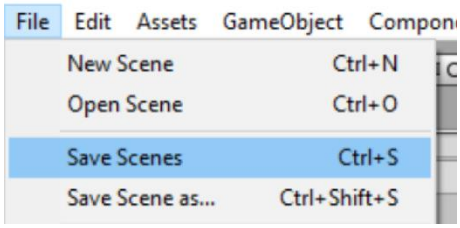
Next we can copy and paste the StartButton object and rename them to Help and Quit. Also change the text of each button to match what it does.



This should produce the following

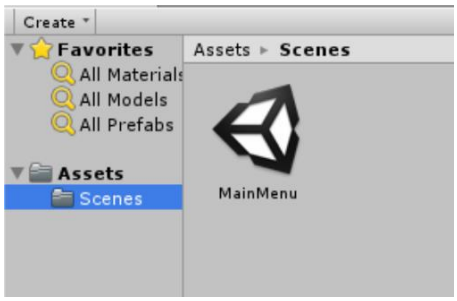


Next, we have to save the scene so that we can access it. Go to the menu, click File -> Save Scenes

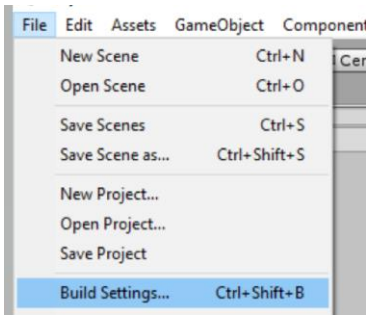


Create a folder in the Assets folder called scenes and save this scene as MainMenu.

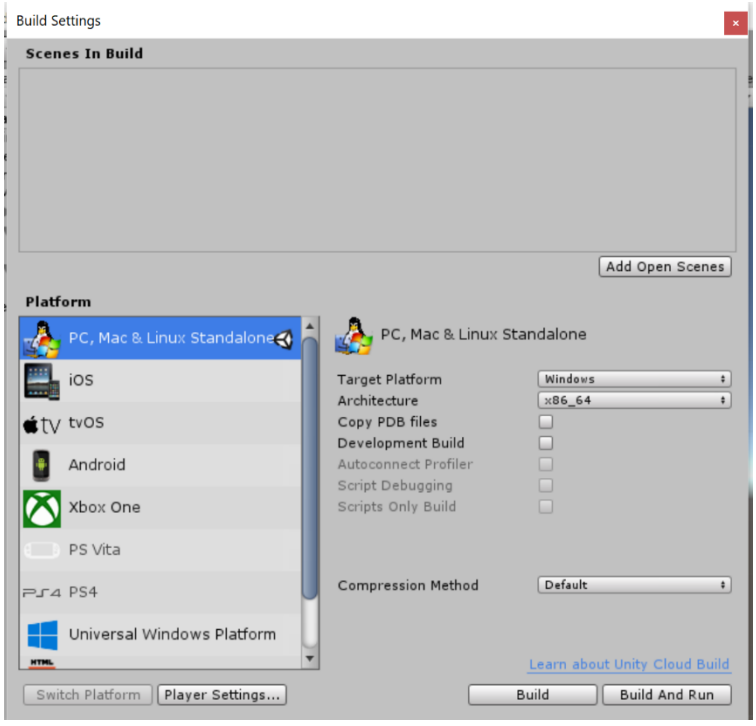
You should end up with the following.



Now go to the Menu and go to Build-> Settings

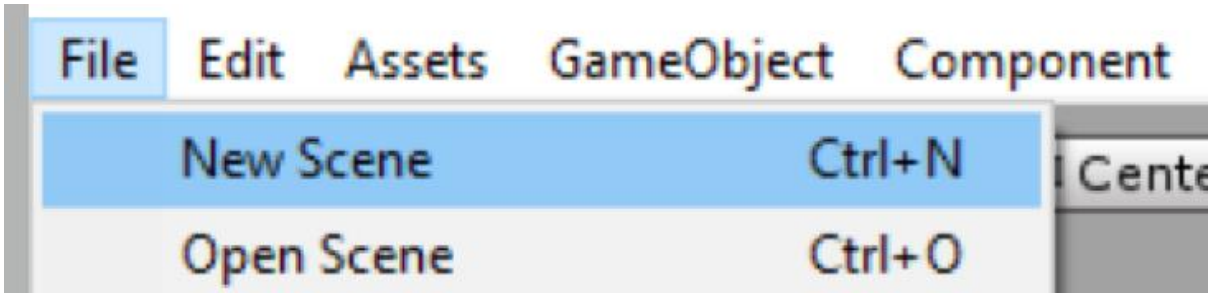


This create the following pop-up that will let us link our scenes.



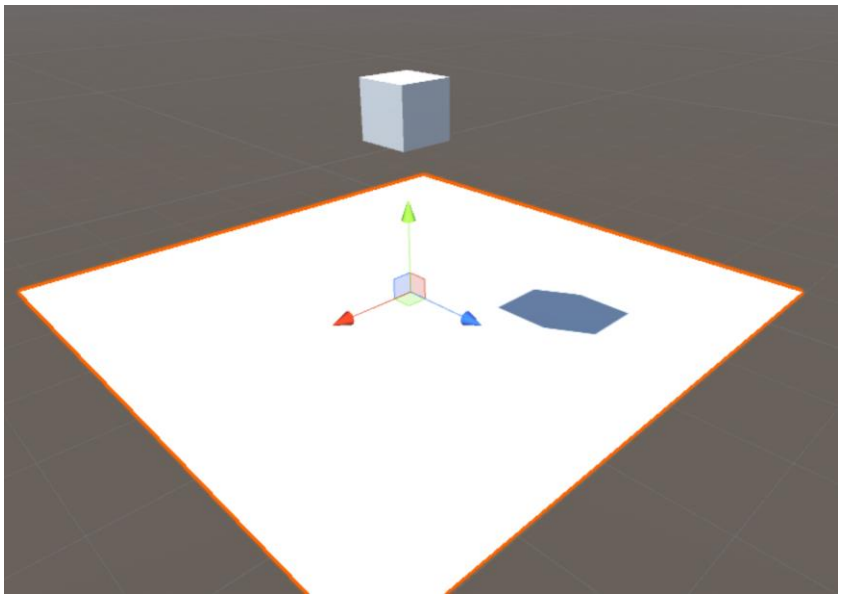
From here click Add Open Scenes, you should see the following appear:

Now that this is listed, note that is has a number of 0 next to it. Next Go File -> New Scene

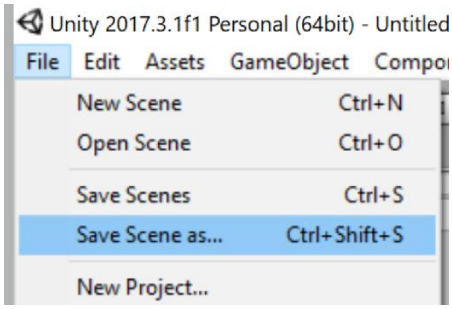


In this scene, switch back to 3D view by click on the 2D icon

Add a plane and a cube to it. Raise the cube above the plane and apply a rigidbody to the cube, so that when the game runs, the cube will drop to the plane.

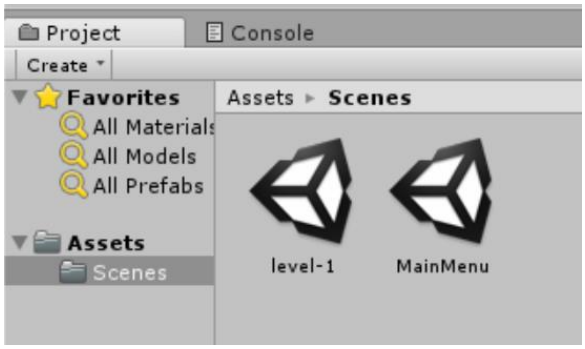


From here go to the menu and do File->Save Scene As. This way you save the scene into the scenes folder.



Name this scene level-1.

You should see the following.

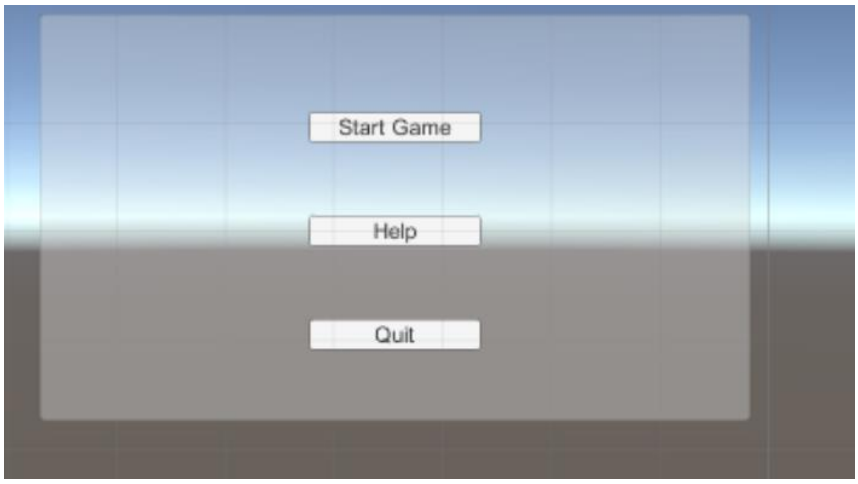


Once again, go in to build settings and add the open scene.

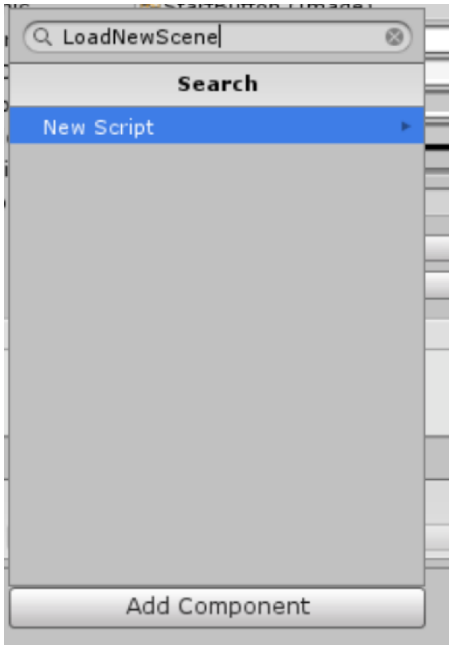
Notice, that level-1 has the index number of 1. This will come in useful for the code we are going to apply to the start game button.

Now open up the Main menu scene by double clicking on it in the assets folder.

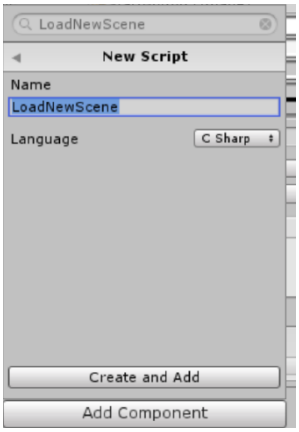
Change back to 2D view, click on Canvas and centre it on the scene so you can se what you are doing.



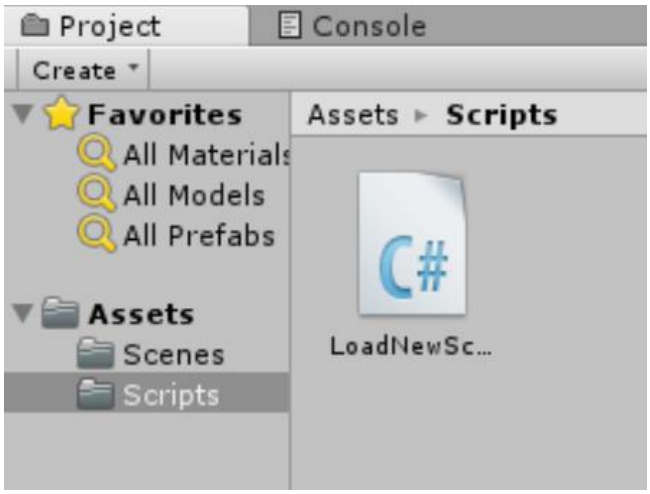
From here, click on the StartButton and add a new script to it.



By putting in an unknown name in the search bar, the add components will automatically figure out that you want a new script by that name, click on new script and then create and add.



In the assets folder create a new folder called scripts and then drag the new script file into it.

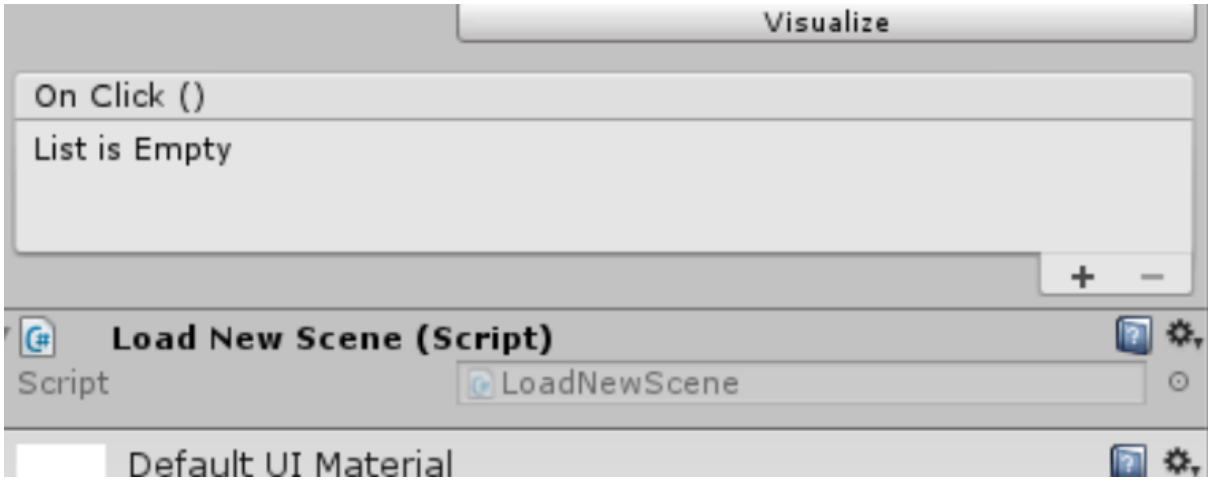


From here load up the script in an editor. And enter the following code.

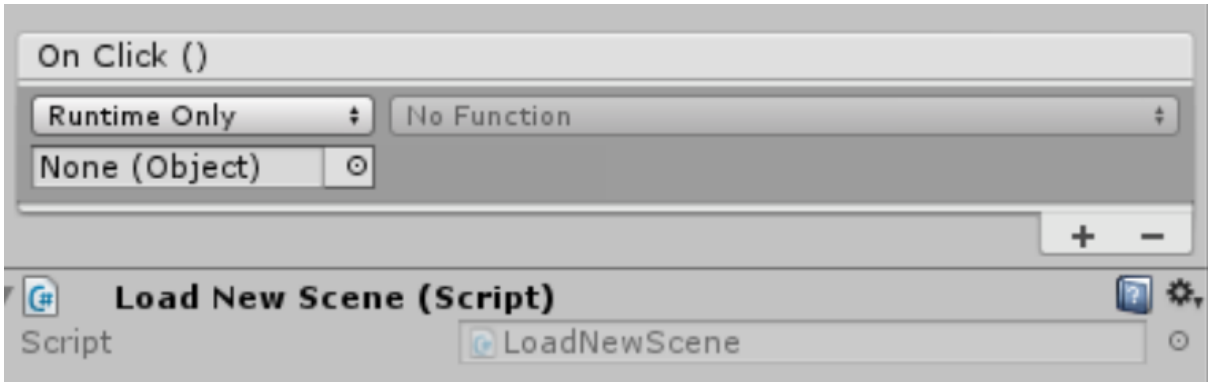
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class LoadNewScene : MonoBehaviour {

    public void LoadByIndex(int sceneIndex)
    {
        SceneManager.LoadScene(sceneIndex);
    }
}
```

Notice the Using line to be added. Once done, save and go back to unity.

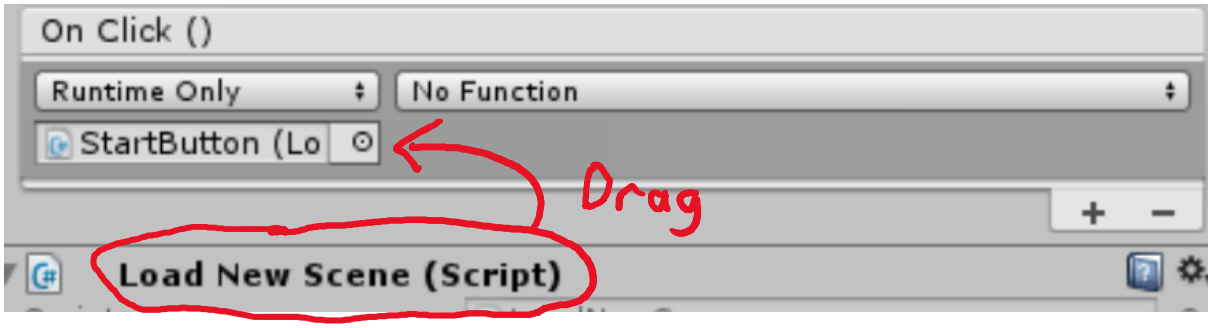
Withteh start button clicked, you should see the following



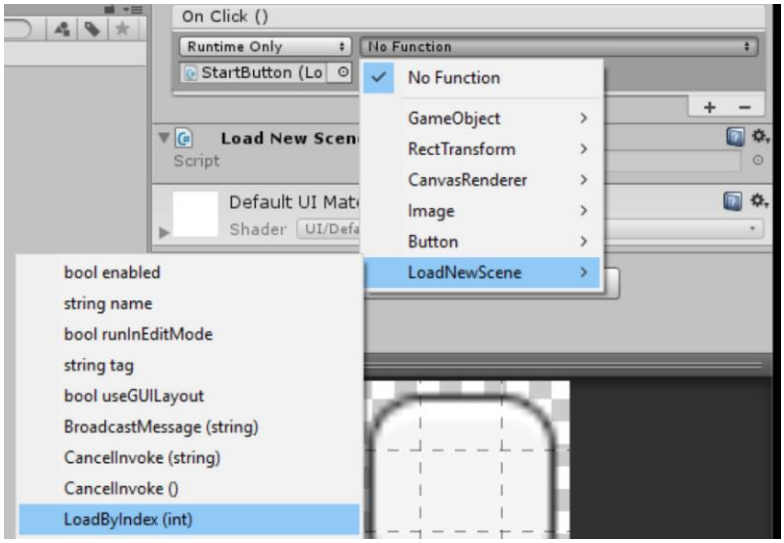
As this is a button we need to add a click event, this event is going to be the script we just wrote, to start with click on the + symbol in the onClick section. You should see the following.



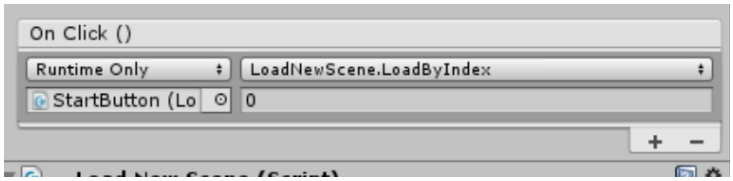
Next, we drag the load New scene onto the None Object setting. It should change like this:



From here we need to be able to offer the scene we want to move too, this is done by selecting the function from the script via the drop down menu. So, click on No Function -> LoadNewScene->LoadByIndex

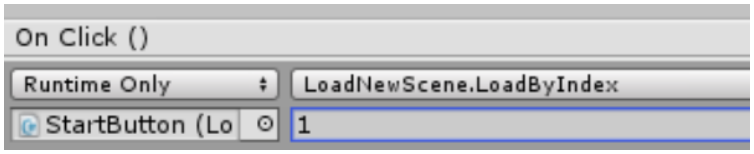


This changes the on click area to this:

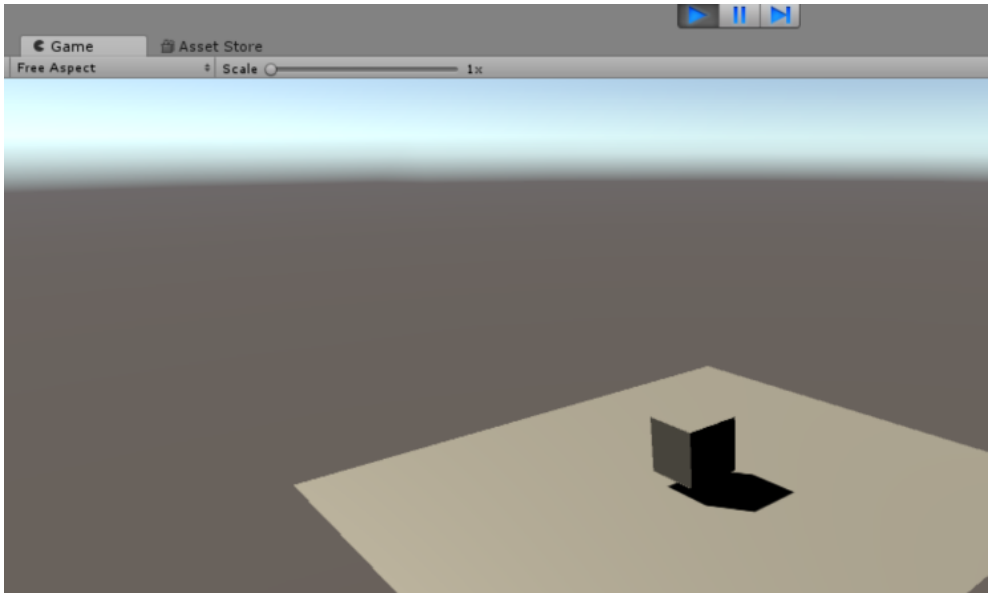


As you can see the parameter it is set to is 0, if you recall from here, we want to click and go to the level-1 scene, it's index number was 1 which was located in the build settings.

So, change the 0 to a 1 and then run it and test the click.

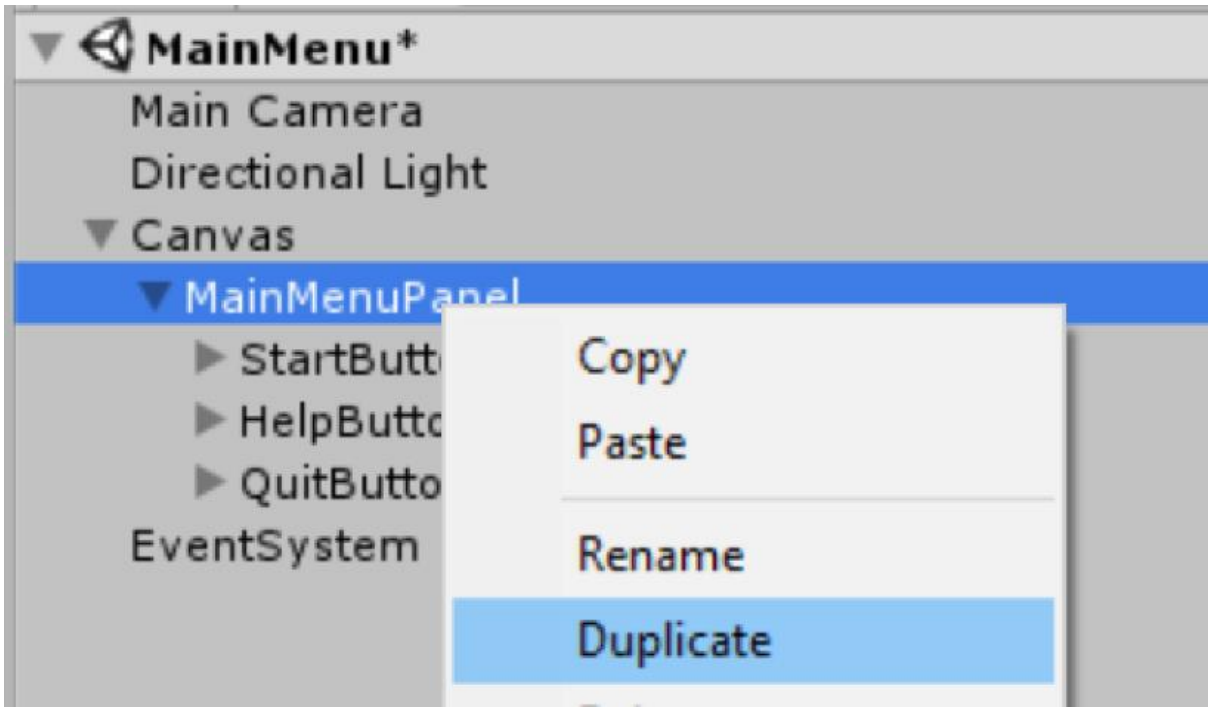


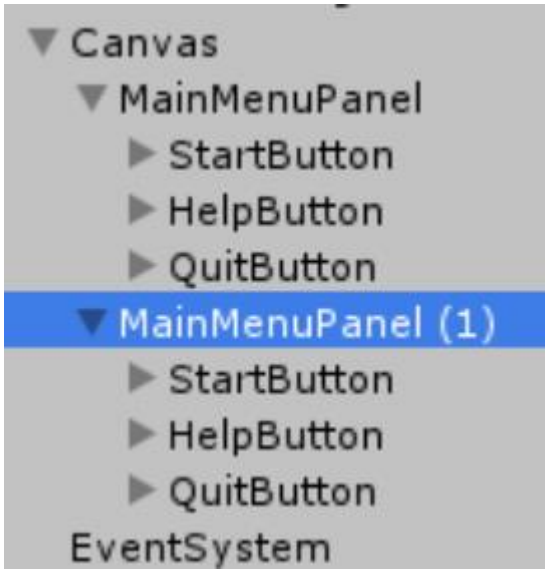
This should transition between the MainMenu scene and Level-1 scene.



Now that this is working, we will look at the help menu item.

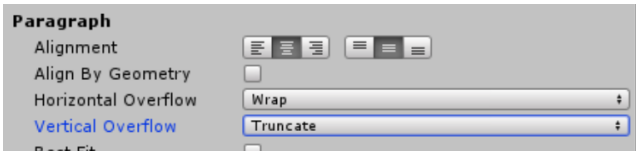
In the hierarchy panel, click on the ManMenuPanel and Duplicate it.



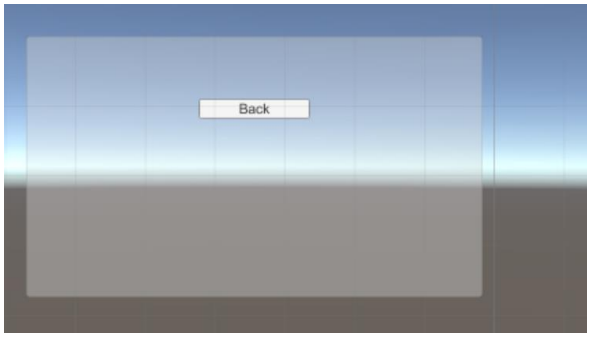


Now, with the Second panel delete the StartButton and HelpButtons. Rename the Panel to Help Panel. Rename the QuitButton to BackButton.

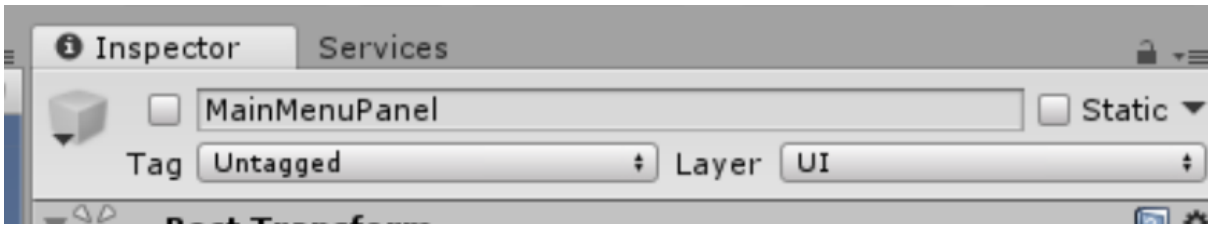
If the word Back is raised from the button, go into the text settings on inspector and change the overflow settings.



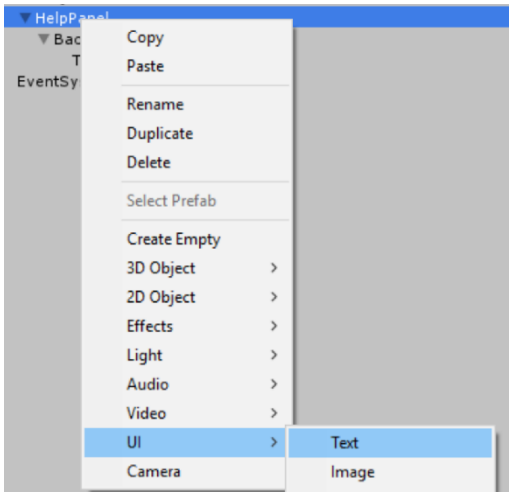
You should see the following.



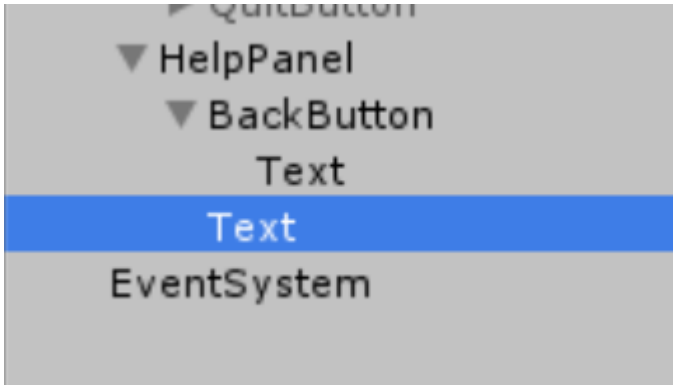
To hide the MainMenuPanel, turn off the active next to it's name.



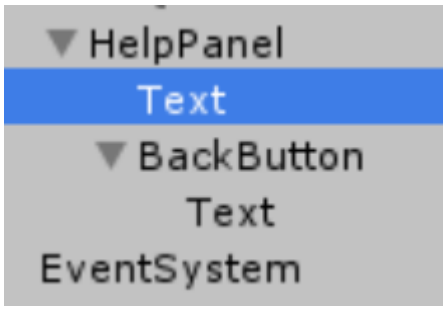
Now, back to the back button, next we are going to add some text. In the hierarchy panel, click on help button and then right click, go to UI->Text



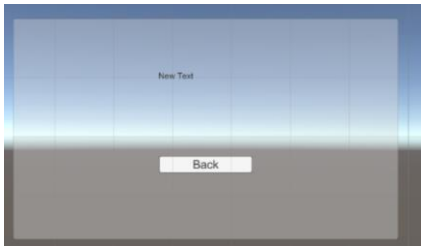
You should see the following



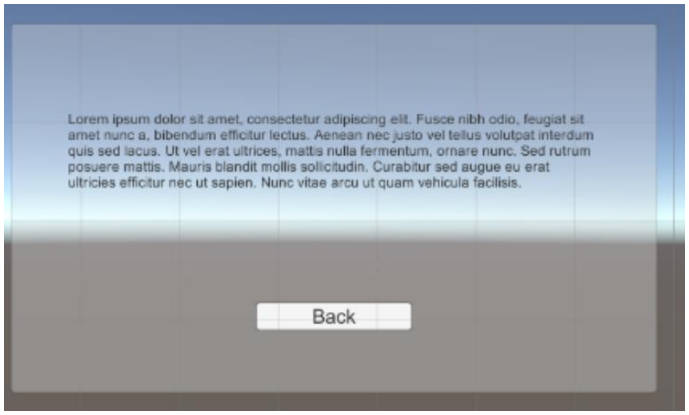
Drag the text above the button.



This will correct the alignment and centre the text on the scene.

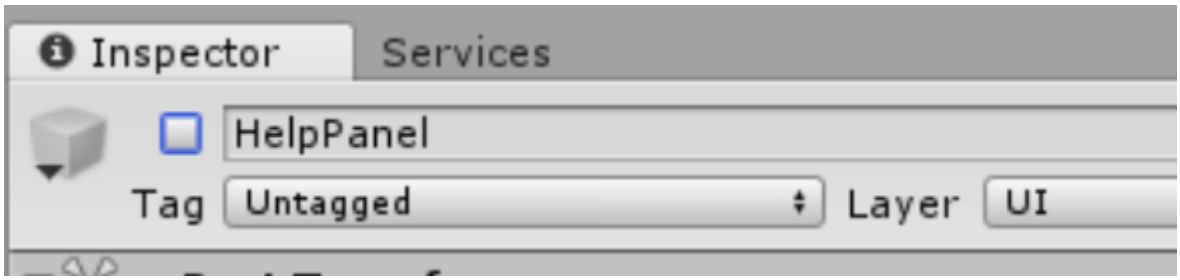


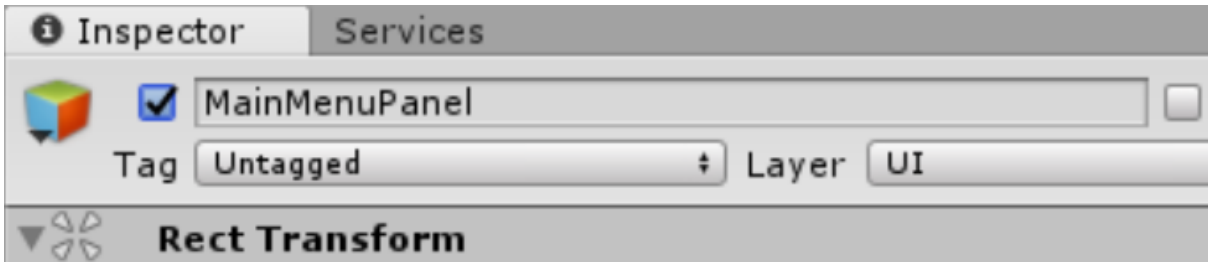
Go into the text settings and add some lorem ipsum text. You should end up with something like this:



Now that the content has been taken care of we will get the button to work. This will be done within unity by swapping the active state of the panels.

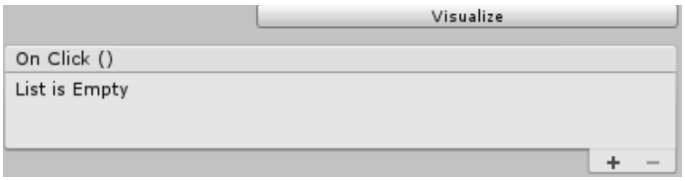
On the help panel, make it non-active and make MainMenuPanel active.



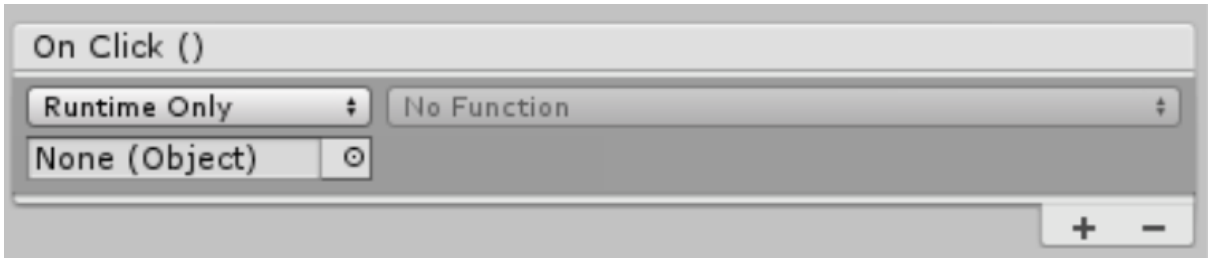


Once this is done one MainMenuPanel, click on the help Button

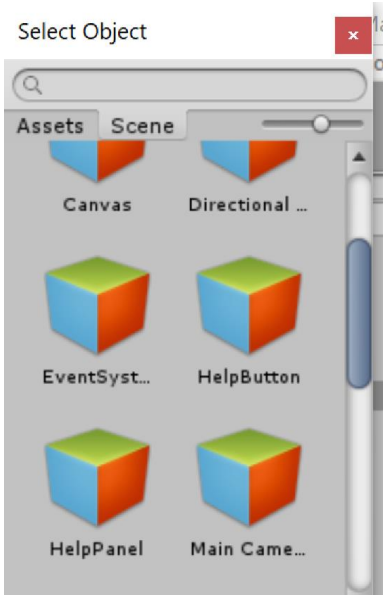
In the inspector panel we have the Onclick area.



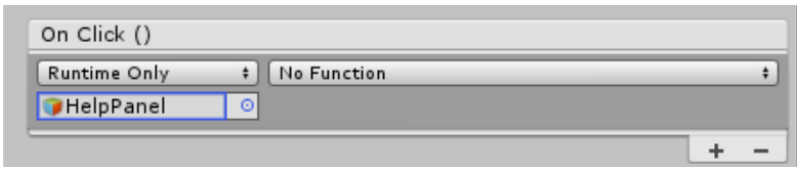
Click the + symbol.



Click on the target icon next to the object, then select the helpPanel



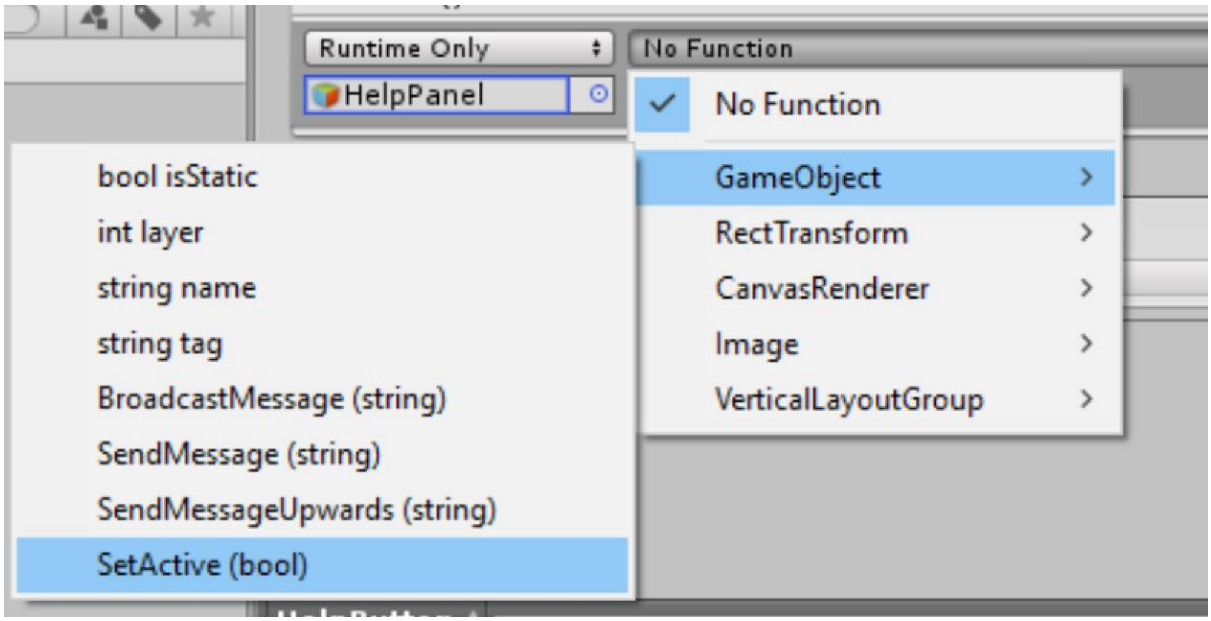
This will provide the following view in the inspector panel.



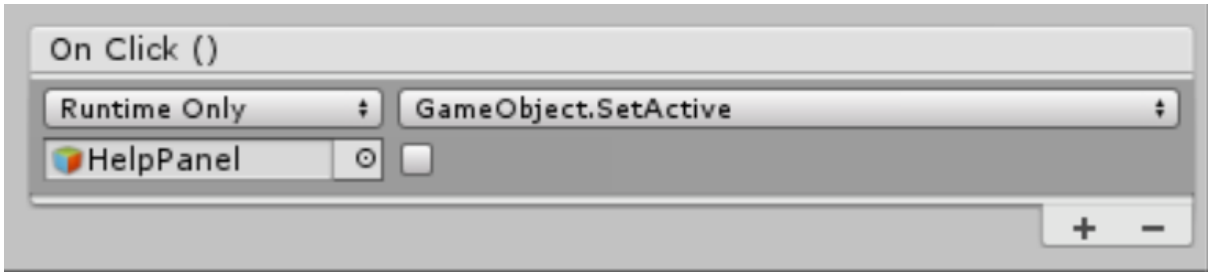
From here we are going to use the No Functions drop down and change the setActive option.

What we are after is When we click on the help button, we want the HelpPanel to become active, i.e. shown to the end user and the MainmenuPanel to become inactive.

To achieve this add the setActive(bool) setting.

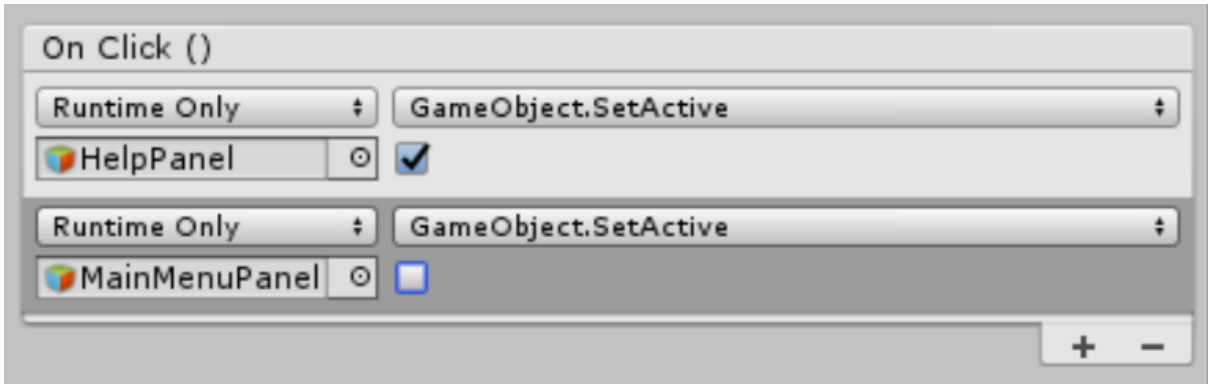


This provides the following:



If the checkbox is not ticked, it means false, i.e. do not setActive. In this case, we want to add a tick, as when we click on the help button we want to see the help panel appear.

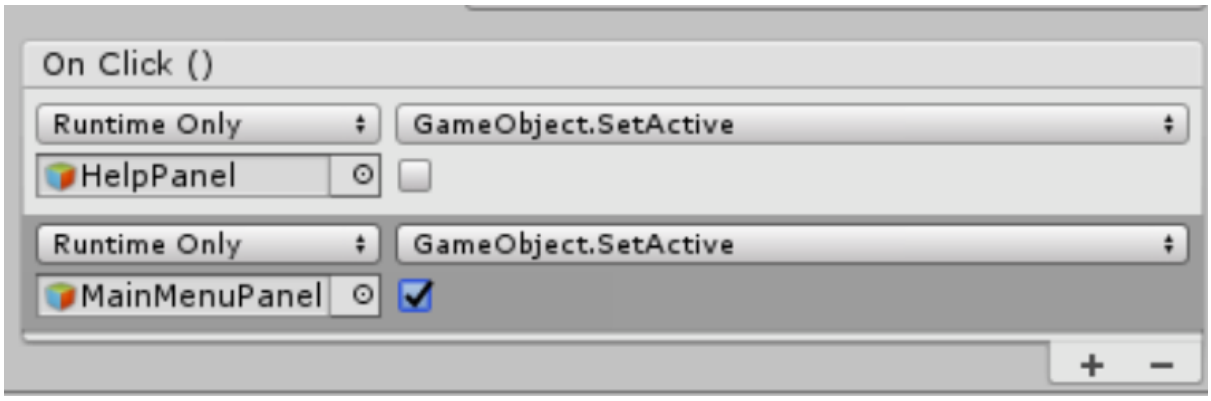
Once you have done that, add the MainMenuPanel to the on click section and set it as not active. You should end up with the on click section looking like this:



Run the program and test it.

This should give you a working menu to the help system. Now, repeat the same steps on the back button in the HelpPanel.

Your Back Button should end up with this:

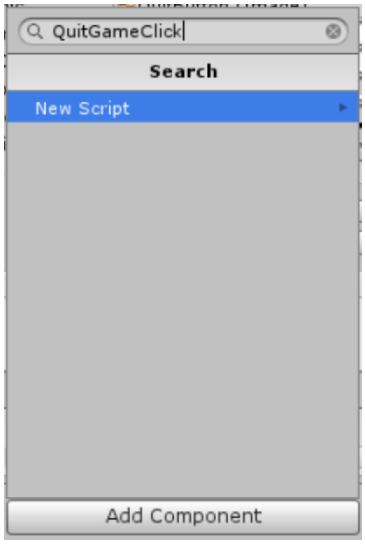


Which is the opposite of the helpButton. Run and test it.

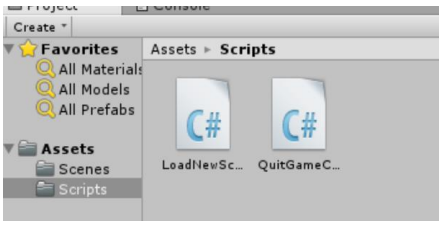
Finally, we need to add the quit functionality to the menu system.

Click on the Quit Button in the hierarchy.

Then in the add Component section, do a search on QuitGameClick; create the new script and go to the code.



Once you have done this, drag the new script into the scripts folder and open up the script to code.

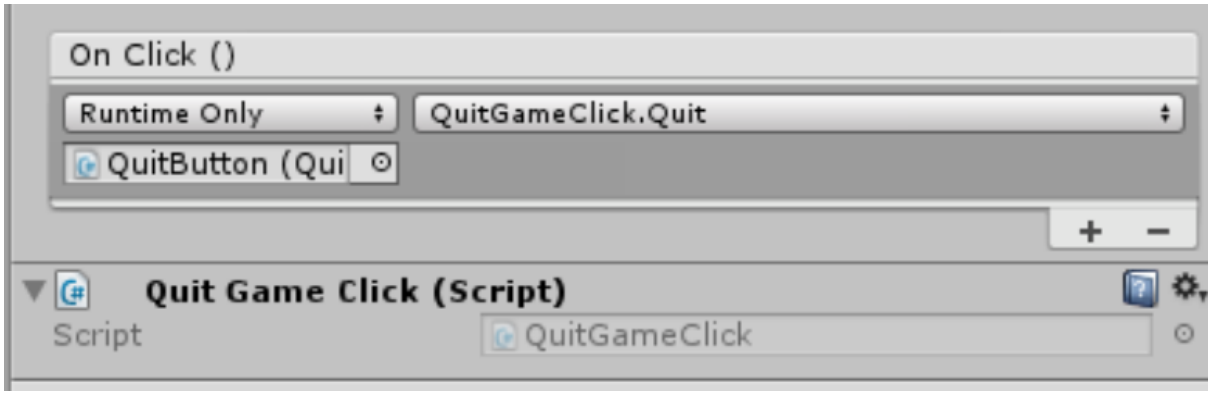


```
public class QuitGameClick : MonoBehaviour {

    public void Quit()
    {
        #if UNITY_EDITOR
                UnityEditor.EditorApplication.isPlaying = false;
        #else
                Application.Quit ();
        #endif
    }
}
```

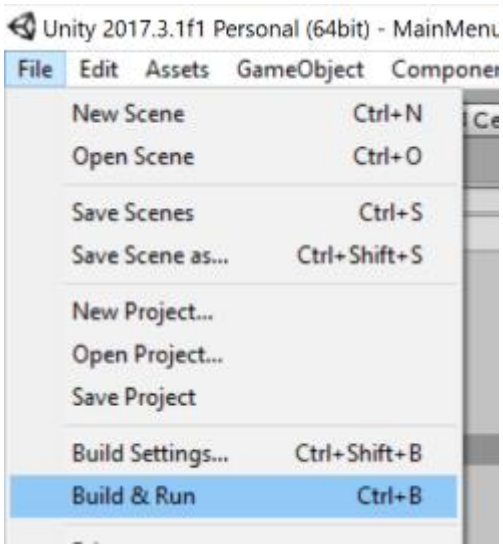
Notice this has the effect of quitting the game once it is used externally of the unity editor and of exiting the unity editor.

Once you have saved this script, we need to assign it to the button onclick event, as we did with the scene manager.

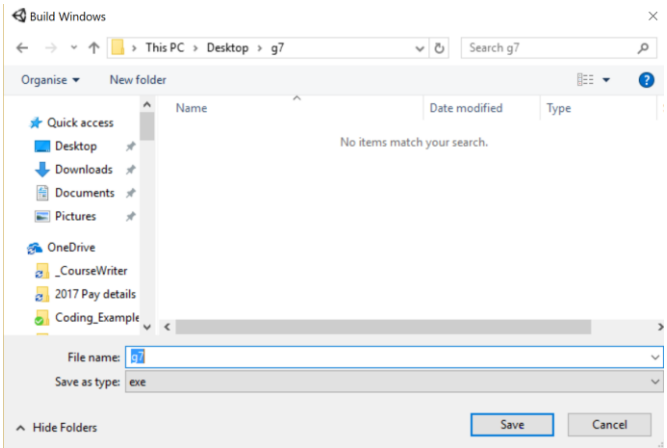
The inspector of quit button should have the following:



To test as an actual game go to the menu and do File->Build & Run



Give the exe a name and save to a location you can find, in this case, I'll use the desktop.



Once it has been built, there should be a configuration window popping open, select the defaults and click play! To test it.