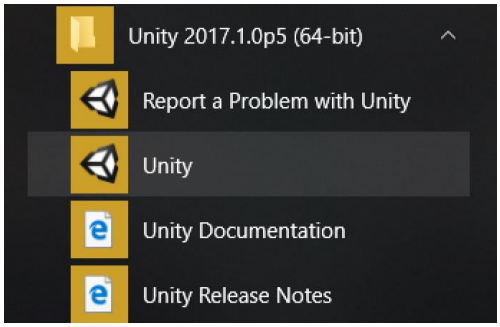
Immersive Environments

Goals:

- Unity – Player movement

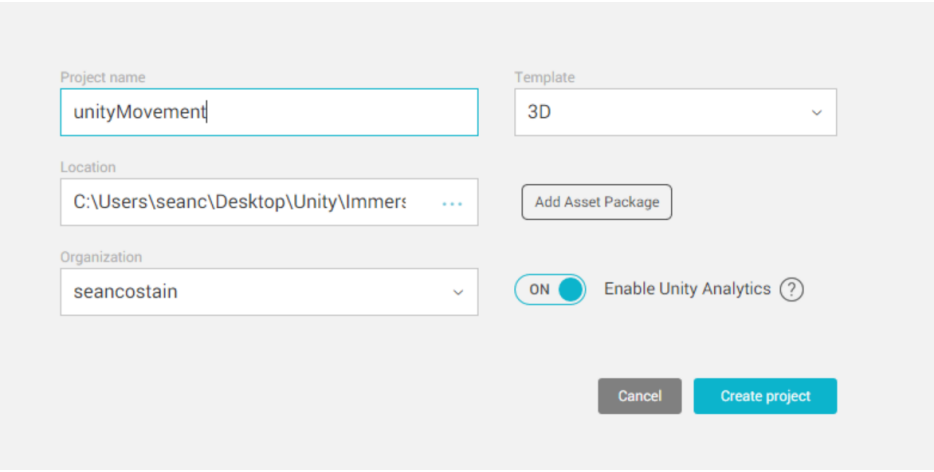
Load up unity



If you don't have an account, you will have to create one. Otherwise sign in with your registered account.

Create a new Project, Click the new icon in the top right.

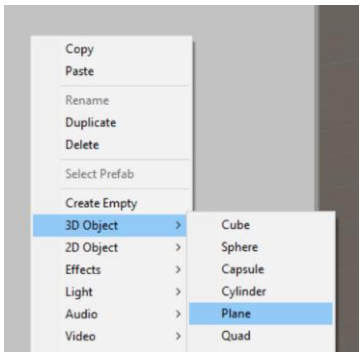
Apply a name and click create project.



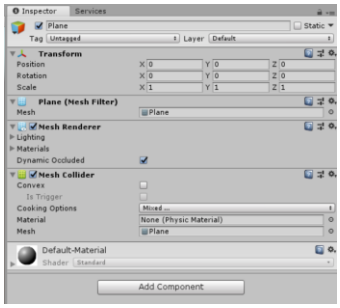
This will open up the following scene

To start with, we will need to create a world to move in. In this case, it is not going to be a complex world, just enough to understand what is going on.

Right click in the hierarchy pane and select 3D -> Plane

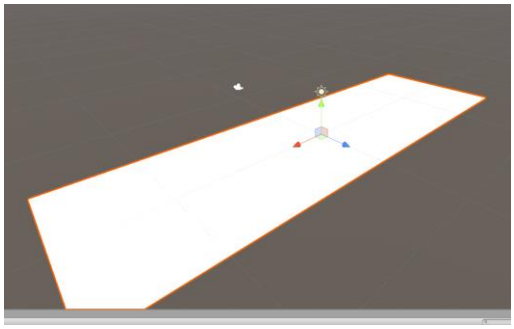


From here, click on the plane and exam it in the Inspector panel.

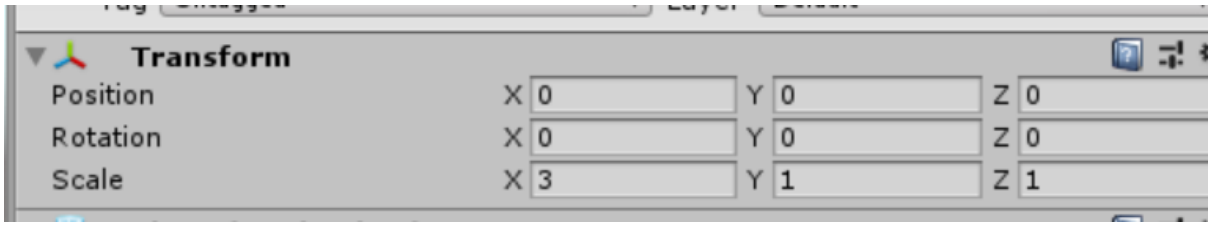


What we want to do is lengthen the plane, this can be done by increasing the scale on the X plane, change the value from 1 to 3.

This should give you the following:

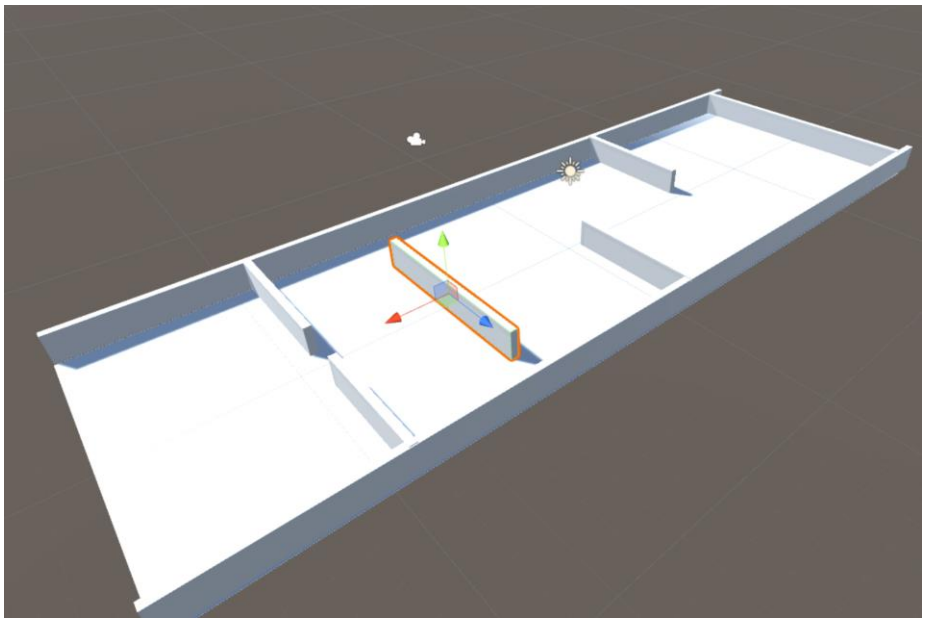


The inspector change is:

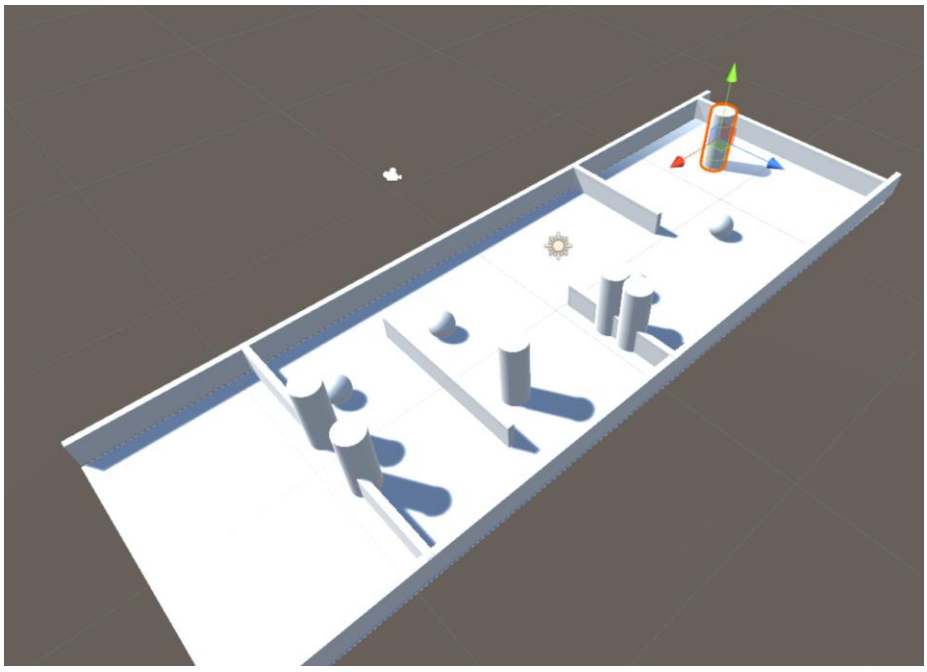


Next add some shaped 3D cubes around the outside and start adding in some walls to run around in.

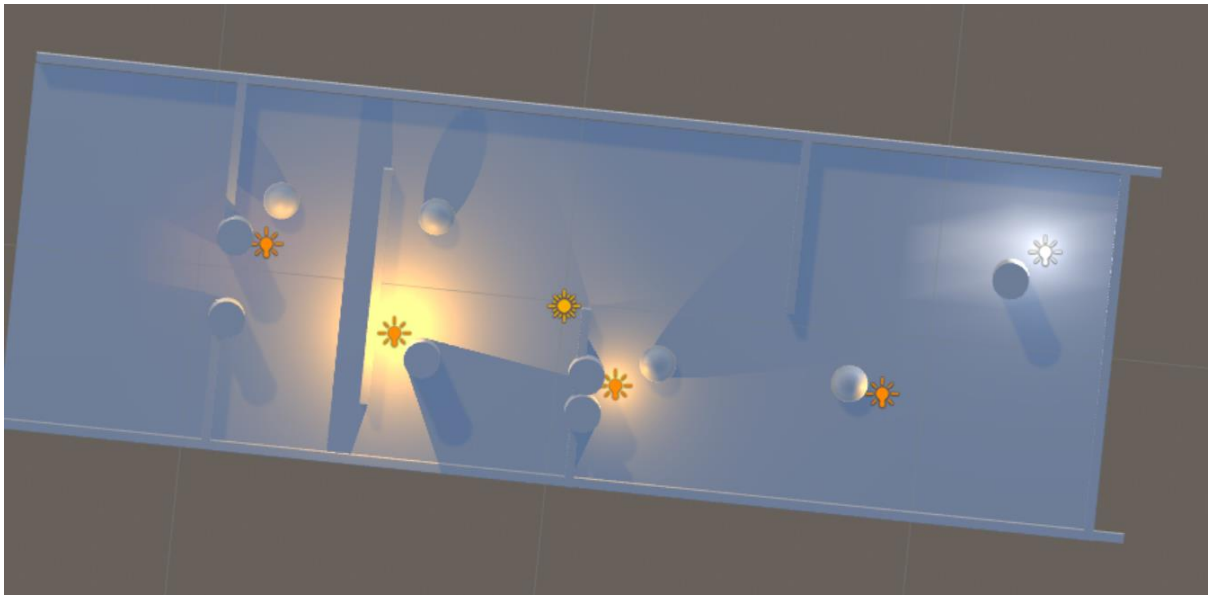
You want to end up with the following looking model; NB it doesn't have to be a perfect match, just something similar.



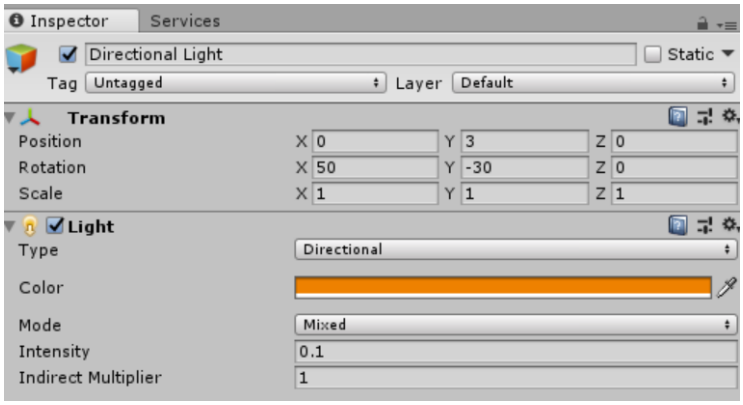
From here, populate it with some cylinders and spheres

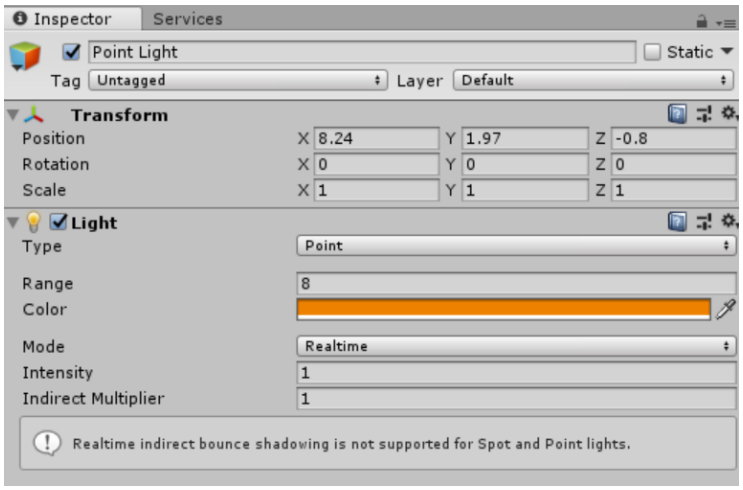


Each of these primitives have just been rotated and scaled to give us an environment to work in. As you can see, we are starting make a bit of a world, now add in some lights of various colours to enhance the world.



These lights are just point lights, with the directional light being dropped to 0.1 in intensity.

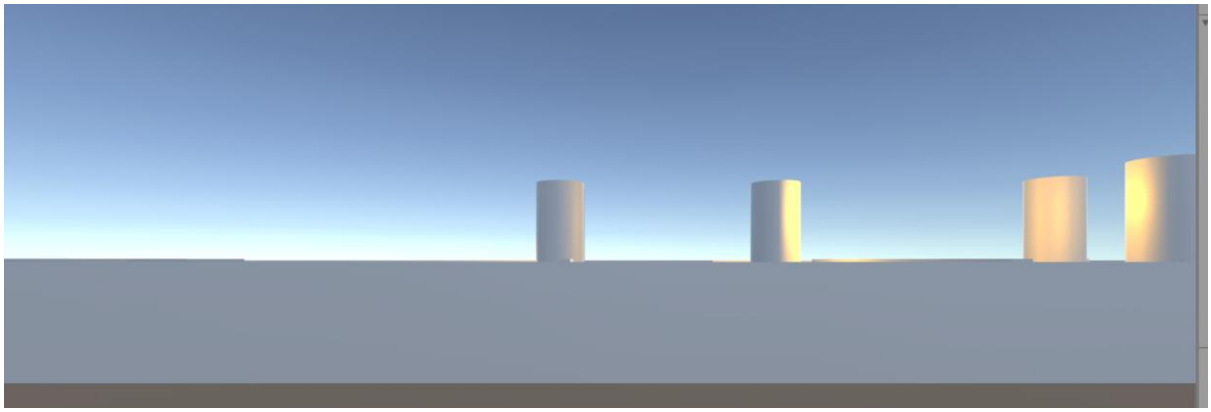




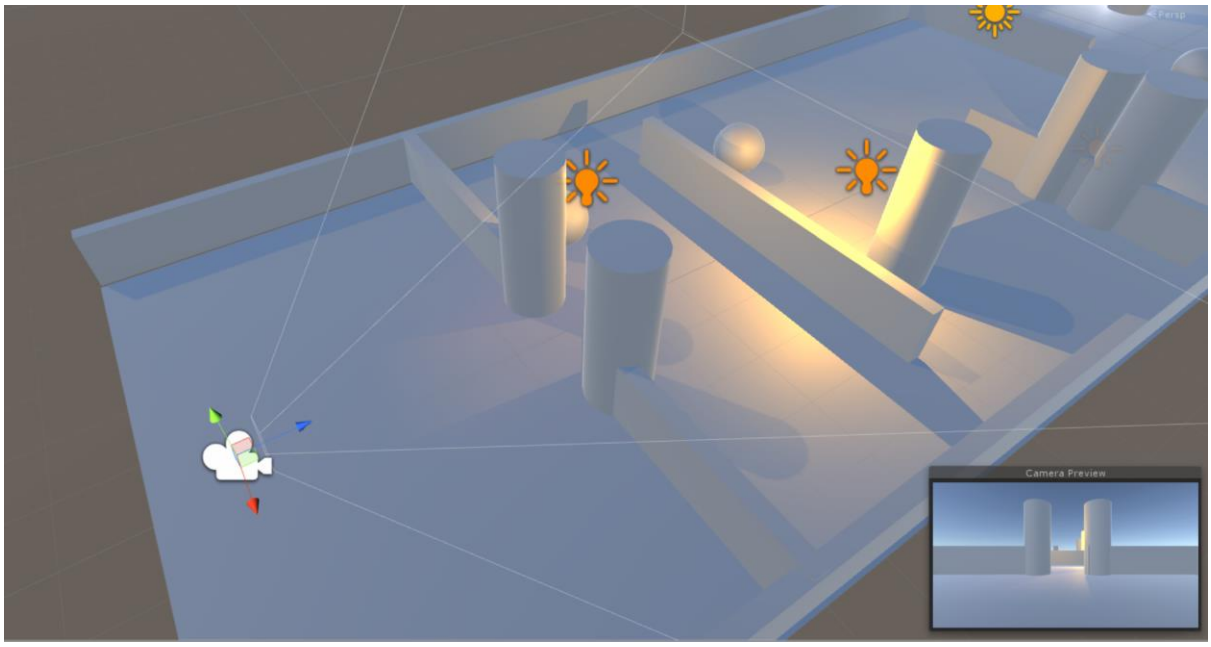
From here, we will now position the camera that we are going to move throughout this environment.

If you click on the play icon in the top

You will see the following



This is because we need to position the camera to our starting point. To do this. Click off play, so you are in the scene view and then position the camera at the start area of our environment.



As you can see in the bottom right, the camera view is shown. This is what we want to see. Though, if we were making a maze/dungeon we would want to increase the height of all of the walls.

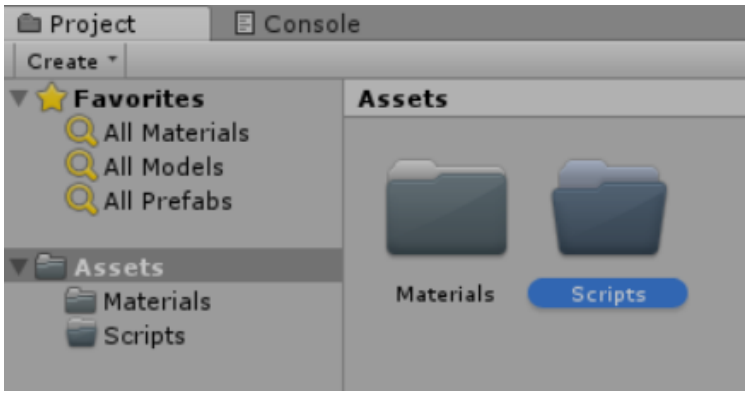
Now that we have done this, we need to add a movement script to the camera.

In unity there are multiple ways to add scripts, we can create the script and then add it to the object, create the script on the object or grab an already script and attach.

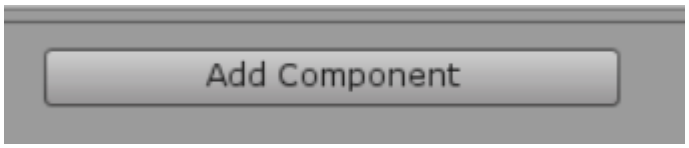
To add the script, we will start by making a folder called scripts in the Assets section, then use the Add Component to create the script. Once this is done, we will need to drag the new script into the Scripts folder.

Add new folder to Assets

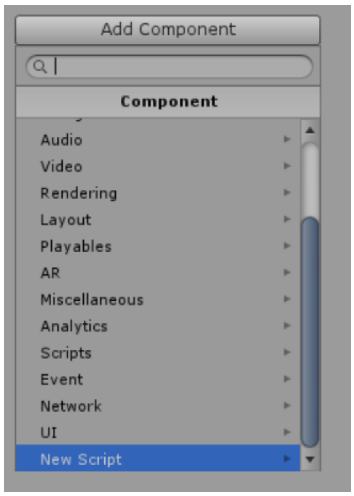
Right click->Create->Folder call it Scripts



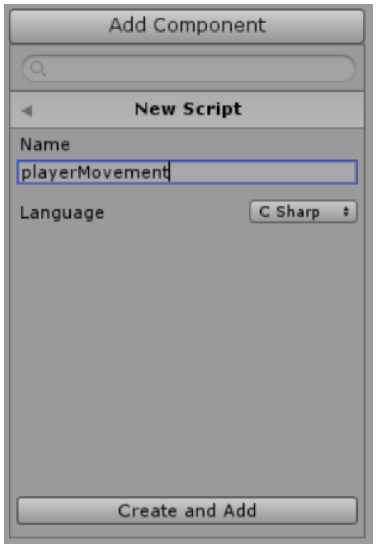
From here click on the Player object in the hierarchy and then go to the Inspector panel and click Add Component.



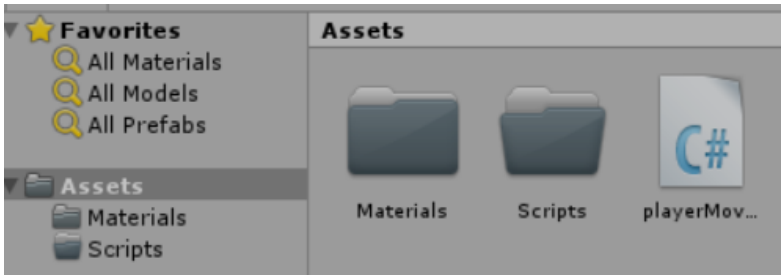
Select New Script



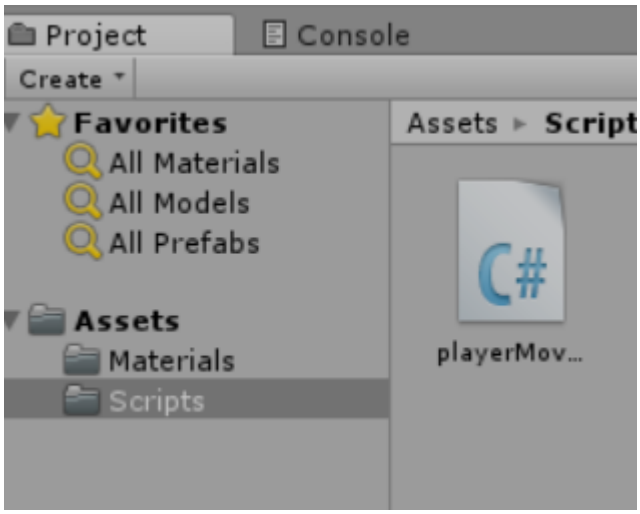
Give the Script a name, in this case it's called playerMovement. We are using the language C Sharp, you'll see it documented as C# sometimes. Then click on Create and Add



Drag the script from the Assets root folder into the scripts folder

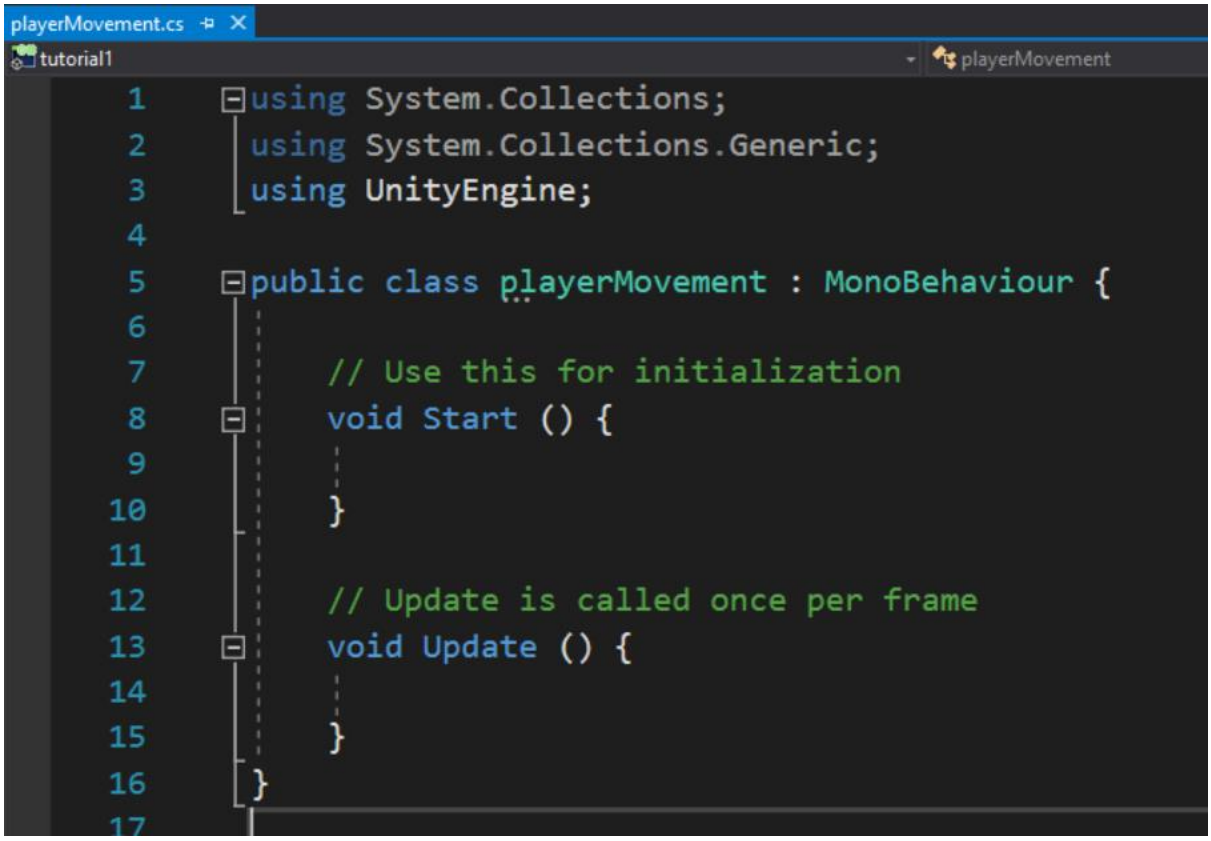


Then go in to the scripts folder



Double Click the newly created script.

It will then open in an editor, this case, the editor used was visual studio 2017. You should see the following:



Add the following code.

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

public class playerMovement : MonoBehaviour {

    // Use this for initialization
    void Start () {

    }

    // Update is called once per frame
    void Update () {
        float vval = Input.GetAxis("Vertical");
        float hval = Input.GetAxis("Horizontal");
        float currentHeight = 0.0f;

        transform.Translate(hval, currentHeight, vval);
    }
}
```

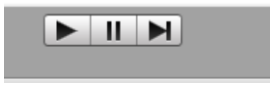
So, what does this all mean?

A float value is a number that has the ability to contain decimals. Vval, hval and currentHeight are all variables, the names could have been anything but it is always best to use names that make sense for ease of understanding. A variable is a memory object that can contain different values but is always referenced by the one name.

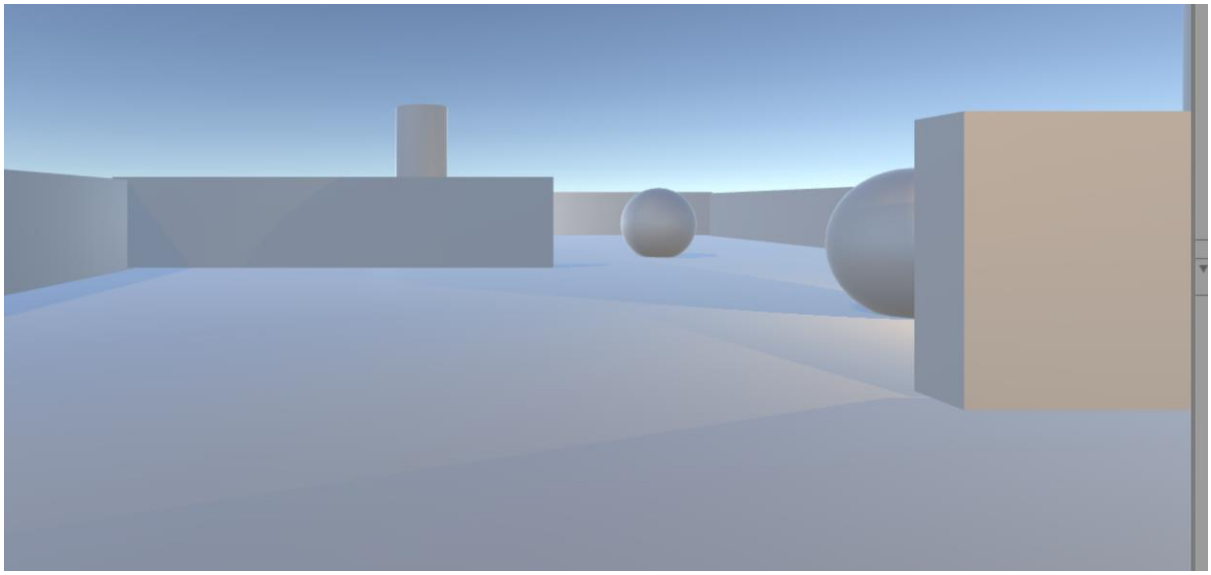
As we are dealing with a 3D world, all objects have 3 axis, X,Y and Z. As such, we need to have values in each axis.

From here, save the script. And go back to unity.

Now you click on the play icon in the top



Then use the W,A,S,D or cursor keys to move the camera through the environment.



On my system, the camera moved very fast. As such, I made a small change to the code. This change is designed to manipulate the speed of the camera.

First, I added a public variable called speed.

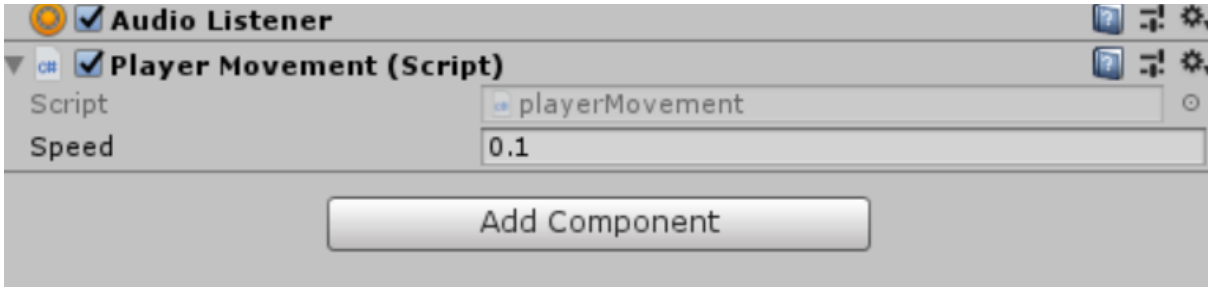
The reason for doing this is that it allows me to manually assign a number to speed until I locate a number that I am happy with, then I can code that speed modifier into the program, in this case it was 0.1.

Secondly, I multiplied hval and vval by speed before the translate was applied, in this manner, a new number was designated and slowed down my camera speed.

The new code should look like this:

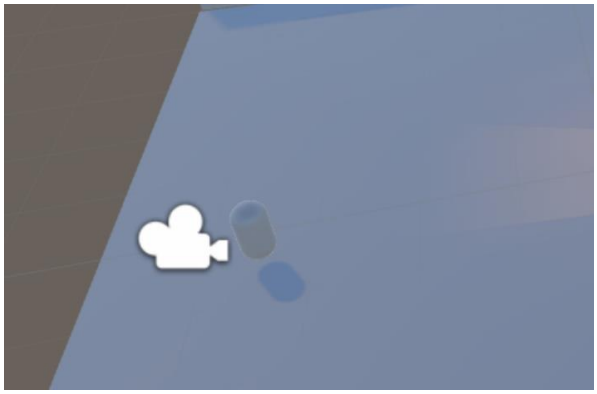
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class playerMovement : MonoBehaviour {

    public float speed = 0.1f;

    // Use this for initialization
    void Start () {

    }

    // Update is called once per frame
    void Update () {
        float vval = Input.GetAxis("Vertical");
        float hval = Input.GetAxis("Horizontal");
        float currentHeight = 0.0f;
        hval = hval * speed;
        vval = vval * speed;

        transform.Translate(hval, currentHeight, vval);
    }
}
```

And if you notice in the inspector in unity, there will be a new box within the script area, in which you can change the number, if you desire.

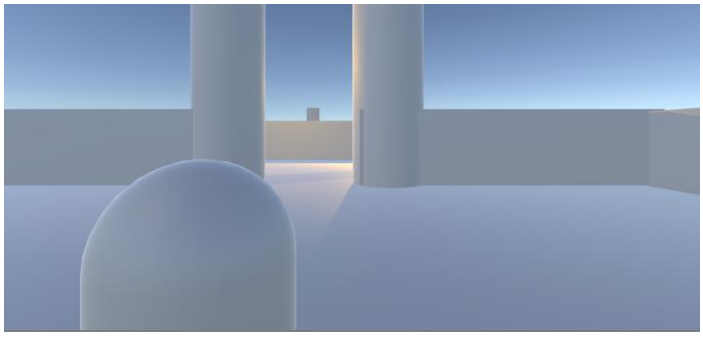


Now that we have the movement around, let's add a player to the camera, so the movement is not always first person, but more third person.

We will use a cylinder as the player. Create a capsule and then position it in front of the camera.

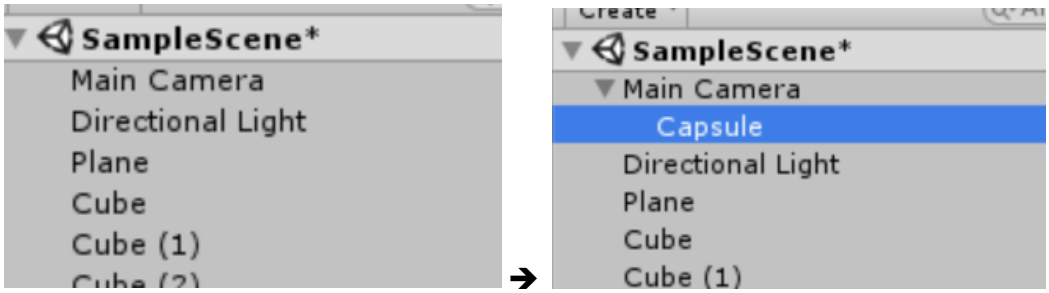


Even though the capsule if floating, from the camera's point of view, it looks perfectly fine.

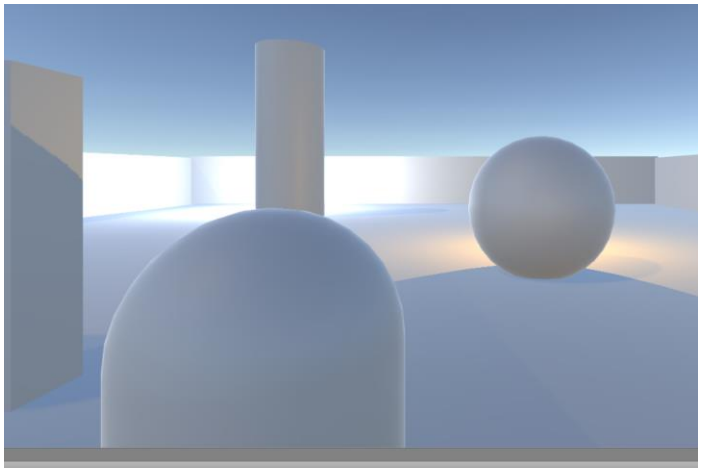


And that is the important thing, to create the illusion that the character is at the right height.

Now, test the game. If you run it up you will see that the camera moves and our capsule stays in the position we started it off. To fix this, we need to turn the capsule into a child of the camera. This can be done by dragging and dropping the capsule onto the main camera.



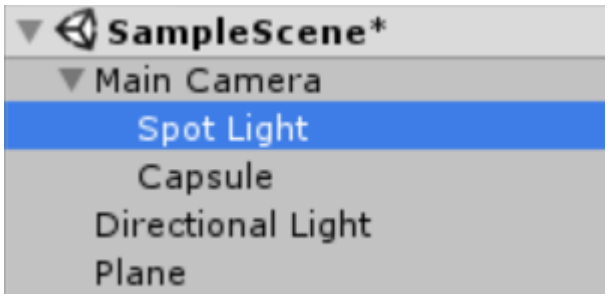
From here, run the game again and test.

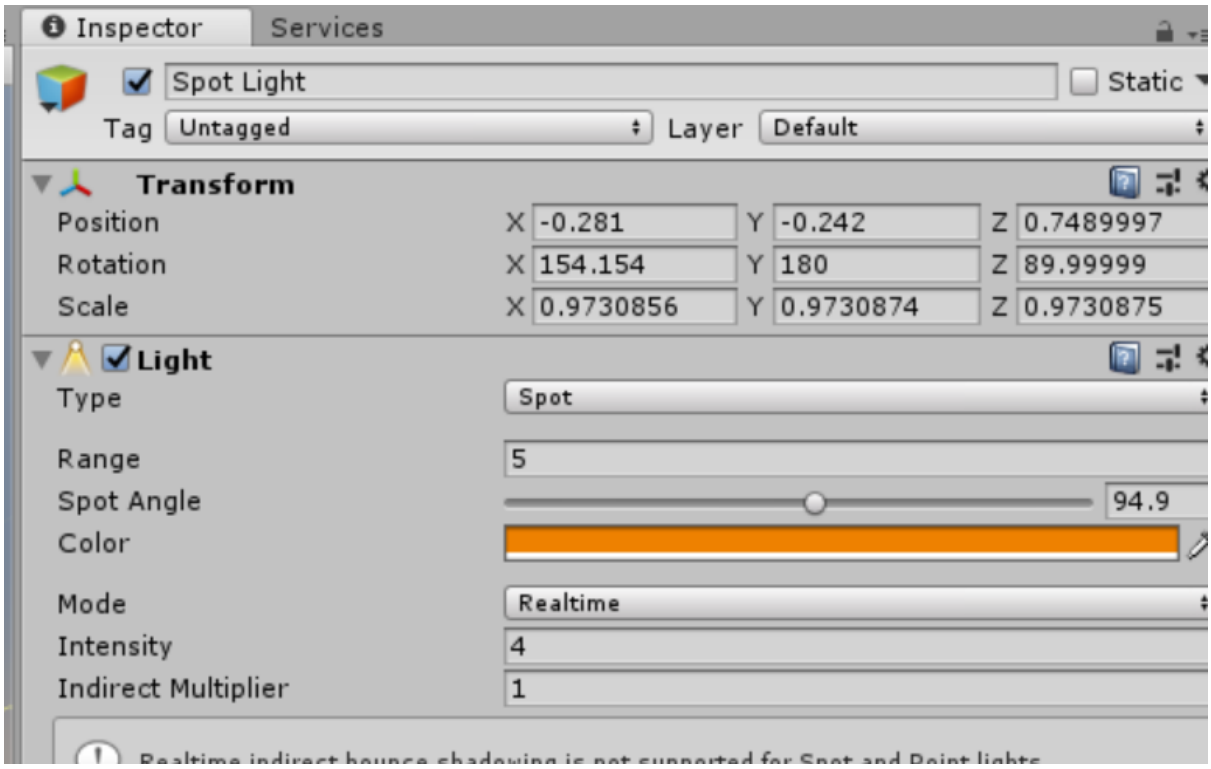


Now when we run around, the capsule player stays with the camera in the world. Next, we will add a spotlight to the character, so they can light the way.

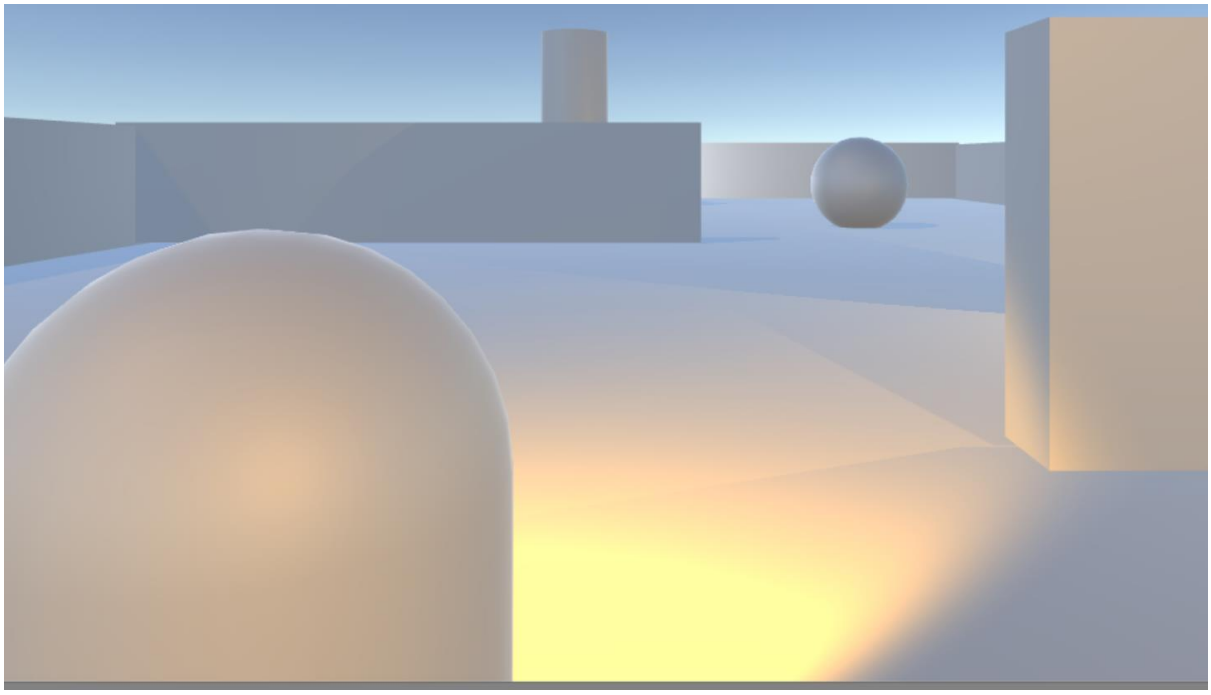
As before, create the spotlight, position it with the player capsule as a child of the camera and then test.

Change colour and range of the spotlight to give it that extra ambience.





This should give you the following view, when walking around the environment.



Just for fun, increase the size of the walls, so the player can’t see the way the maze/dungeon looks.