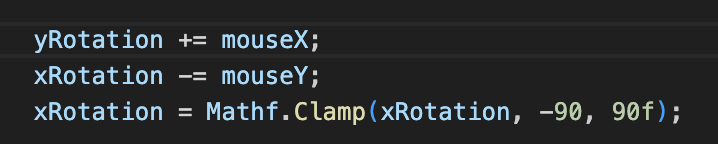
**Learning journal - Inayah Iqbal**

**Stopping excessive camera rotation: Camera max and min limits**

* I had a problem where the player camera was able to move 360 degrees in all directions, which is undesirable as it causes clipping and strange camera angles.

Solution: To stop the player from rotating the camera too much, making it appear buggy since it goes through the player character, I realized a simple solution was to add bounds of –90 and 90 degrees in my player camera script like below,

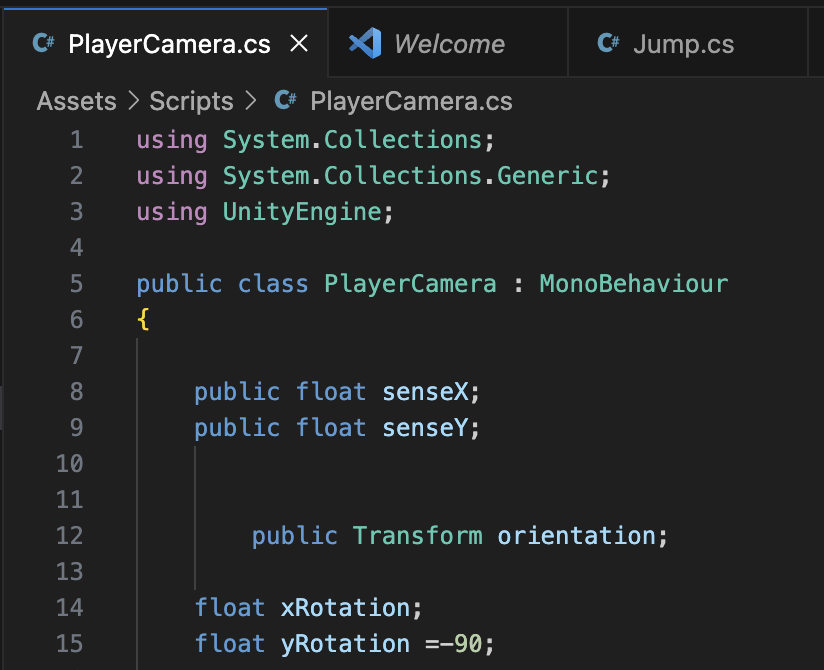
The 3rd line of code uses ‘clamp’ and then inserting the values you would like will cause the camera to clamp at these angles.

**Player camera is not facing the correct direction on begin play**

* I had this problem where no matter how much I rotate the player camera it doesn’t change the camera’s view to face the correct direction on begin play. Note: because my player character (cylinder) is parented to an empty with 2 other empties such as ‘orientation’ and ‘camera position’. I experimented with rotating these as well in case they caused a change in the camera angle, but doing this had no results.

Solution:

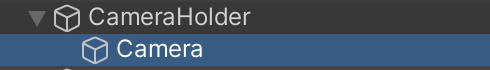
In my code I needed to type in the actual value I wanted to camera to face, in my case it was ‘-90’ in the x axis. (please scroll down)

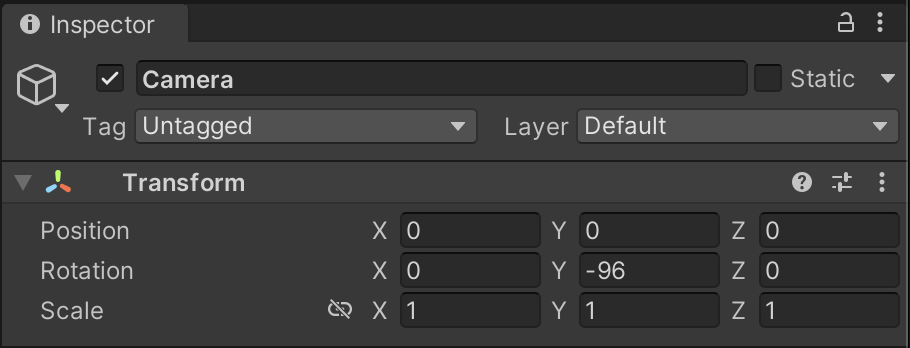


On line 15, I added =-90; to float yRotation



I also had to update my camera’s transform value:

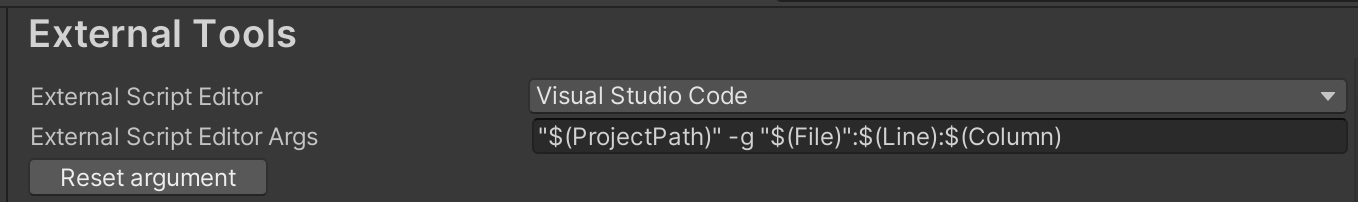




Y rotation = -96, make sure the position is 0, 0, 0, for XYZ so it stays at the world origin where the player is.

**Visual studio scripts not being opened in the correct application**

* I had this issue where when starting up my unity project on a different desktop computer, when trying to access my scripts, they would open in adobe Dreamweaver rather than visual studio. The solution was to open the unity settings and in eternal editors choose external script editors and make sure it’s selected to ‘visual studio’.



**Changing / editing my map (making maps that are playable)**

* A problem I had when trying to design my map was the scale of it wasn’t large enough. My game uses force whenever the player moves which makes them launch very far. I had to do find problems with the map and test it out to see where I was constantly failing so I could change some things around to create a map that was playable and could be completed.

Solution: I edited different parts of the map as I played, when I found that I was falling down through the map, I knew that the difficulty was too high, and I needed to make it easier by making the platforms longer as well as adding more to jump on. I also needed to keep the platforms more spread out than I had originally intended.

**Handling different Unity versions (changing project versions)**

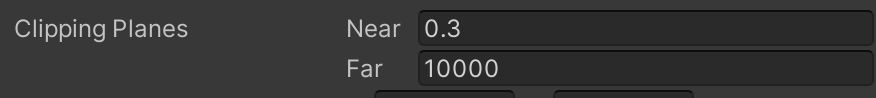
* When I first started creating my game project it was on an older version of unity and I needed to update it.

Solution: Before updating I made another copy of my project, after installing the new version of unity I just opened it like usual and saw if anything was broken. Alternatively, if that doesn’t work, I can also create a new unity project in the new version of unity and then just drag in the assets folder from my original project. But I also learnt that going down project versions may cause projects to break so keep this in mind before changing down versions.

**Map disappearing when going too far out**

* An issue I had was causing the map to disappear or become invisible when I jumped away too far, this made it hard for the player to see where they were and where they should be going.

Solution: In my main camera’s inspector, I can edit an option called ‘clipping planes’ and change the ‘far’ value to something line 10,000 so that when I move far away from my main map it will still be visible.

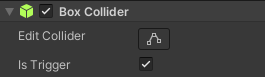


Clipping planes are used to help hide geometry far away from the camera to help with efficiency, specifically if you have a very large map, so it doesn’t all need to load at once.

**Scripts not executing using box collisions**

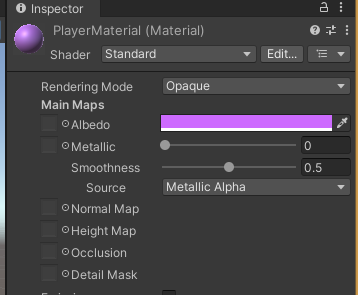
* An issue I had was I wanted to trigger an event when the player entered the box collision specifically, but nothing seemed to happen when the player entered the box even though the script was added as a component to the collision in the inspector.

Solution: in the inspector of the collision there is an option called ‘Is Trigger’ you MUST remember to check this otherwise the scripts will not be executed.



**Making materials**

* I wanted to change the default colors of different shapes and planes within my game to color code them. To do this you have to create a new material by right clicking in the browser and selecting ‘new material’ this will create a new editable material where you can select the color and drag and drop this same material on as many different assets as you like, there’s no need to duplicate the same materials for each object.



**Button not pressing**

* I had this issue where when I coded the script for a button which reset the game, I couldn’t press the actual button to see if it worked.

Solution: In one of my scripts called ‘PlayerCamera’ and I had 2 lines of code that locked and made the cursor not visible.



I decided to make sure these 2 lines did not compile. After I also renamed my scene from ‘samplescene’ to ‘main’ just to stop any errors happening when I import any new scenes into my project, I learnt that this is good practice.

Next in my ‘PlayAgainButton’ script I changed the last line of code from sceneManager.LoadScene(‘SampleScene’); to just sceneManager.LoadScene(0);

This utilizes the index number instead of the name of each level, making it better if I decide to rename my levels or add new levels. This fixed by button to make it start working when I tried clicking it.