CW1

Games Engines

24678805@edgehill.ac.uk | 13/11/2020

Coursework one

Callum McLaughlin

Contents

[**Task 1- Unity: The Basics** 2](#_Toc56099618)

[**A guide to getting started in Unity** 2](#_Toc56099619)

[**Introduction** 2](#_Toc56099620)

[**How to find and Install Unity** 2](#_Toc56099621)

[**Creating a Unity Account** 4](#_Toc56099622)

[**How to set up a Unity3D Project** 5](#_Toc56099623)

[**Changing default editors** 6](#_Toc56099624)

[**Layout Options** 7](#_Toc56099625)

[**The Asset Store** 9](#_Toc56099626)

[**Task 2 – Environment Design** 10](#_Toc56099627)

[**Task 3 – Character Controller** 11](#_Toc56099628)

[**Task 4 – Collision Detection** 13](#_Toc56099629)

[**Appendices** 14](#_Toc56099630)

[Assets Used (All from the Unity asset store) 14](#_Toc56099631)

[Research 14](#_Toc56099632)

[Ref. 1 14](#_Toc56099633)

[Ref. 2 14](#_Toc56099634)

[Scripts 15](#_Toc56099635)

[PlayerMove (1) 15](#_Toc56099636)

[PlayerMove (2) 16](#_Toc56099637)

[PlayerLook 17](#_Toc56099638)

[FinishCollision 18](#_Toc56099639)

[WallCollision 18](#_Toc56099640)

[FloorCollision 19](#_Toc56099641)

# **Task 1- Unity: The Basics**

## **A guide to getting started in Unity**

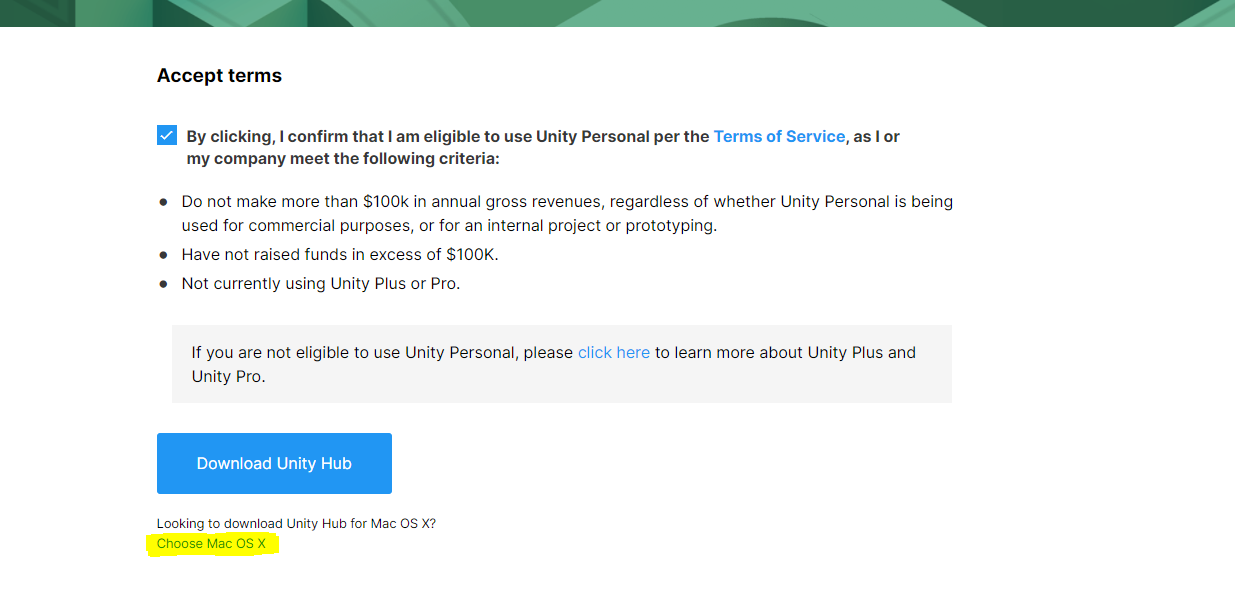
### **Introduction**

Unity is a games engine that is capable of facilitating the creation of both 2D and 3D games. It supports game development for a number of platforms including PlayStation, windows, iOS, Oculus Rift and many more.

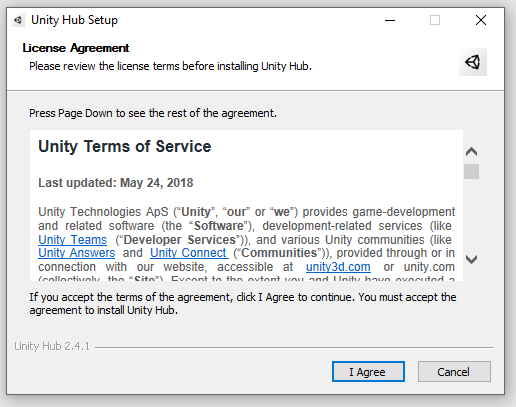
This short guide will get you started in Unity and show you how to download and install Unity on windows. Although this guide will not show specifically how to download unity for macOS, the process is very similar, and the guide can still be used to follow a macOS download.

### **How to find and Install Unity**

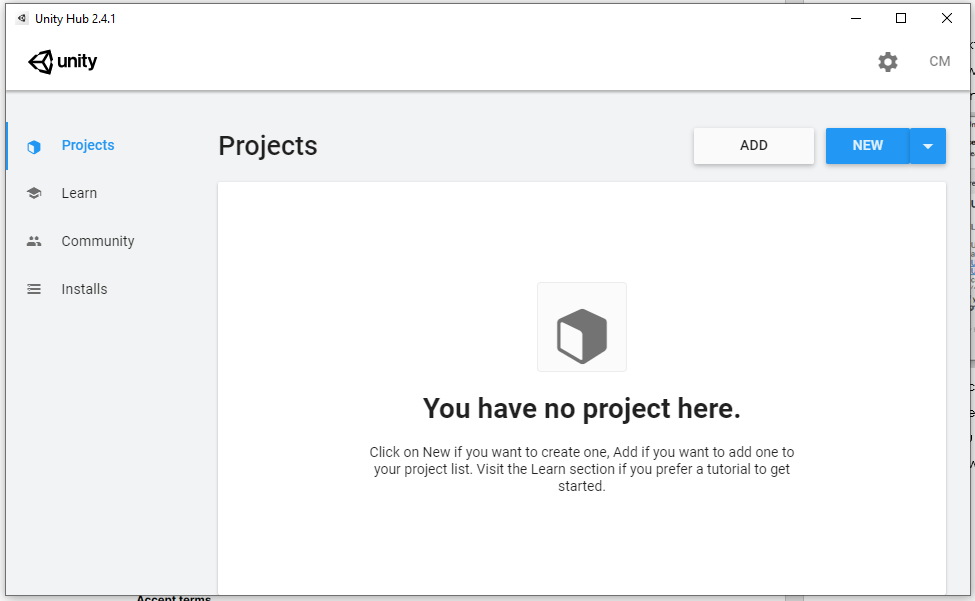
1. Navigate to store.Unity.com/download in a browser of your choice, accept the terms and conditions if applicable and click “Download Unity Hub”. If using macOS, click on the link below the download button that says “Choose Mac OS X” as shown below.



1. Next, a download will start in your browser called “UnityHubSetup.exe”. When the download is completed, left click on the download to run the installer. You will then see a window like the one shown below.

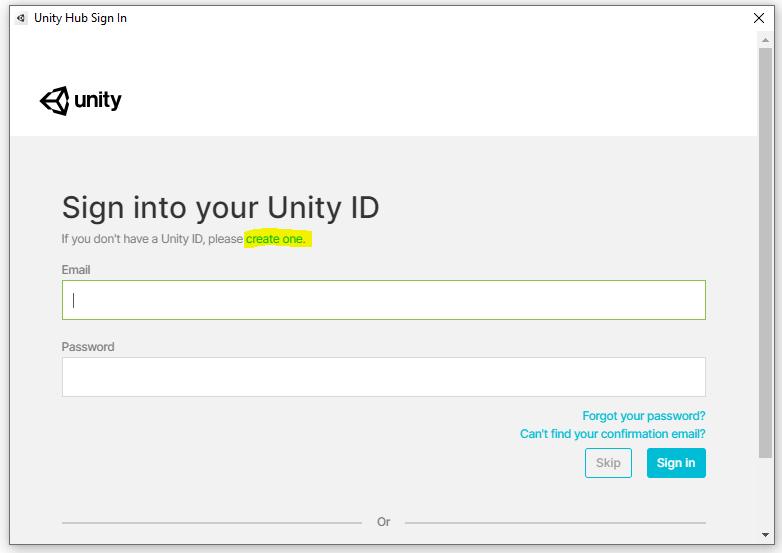


1. Accept the terms if applicable and click the install button on the next screen. After the installer has finished, tick the “Run Unity Hub” checkbox, and click finish.
2. You may then be asked to allow Unity access through the windows defender firewall, if so, click the “allow access” button to proceed.
3. Unity Hub will then launch, and you will see an interface like the one shown below.

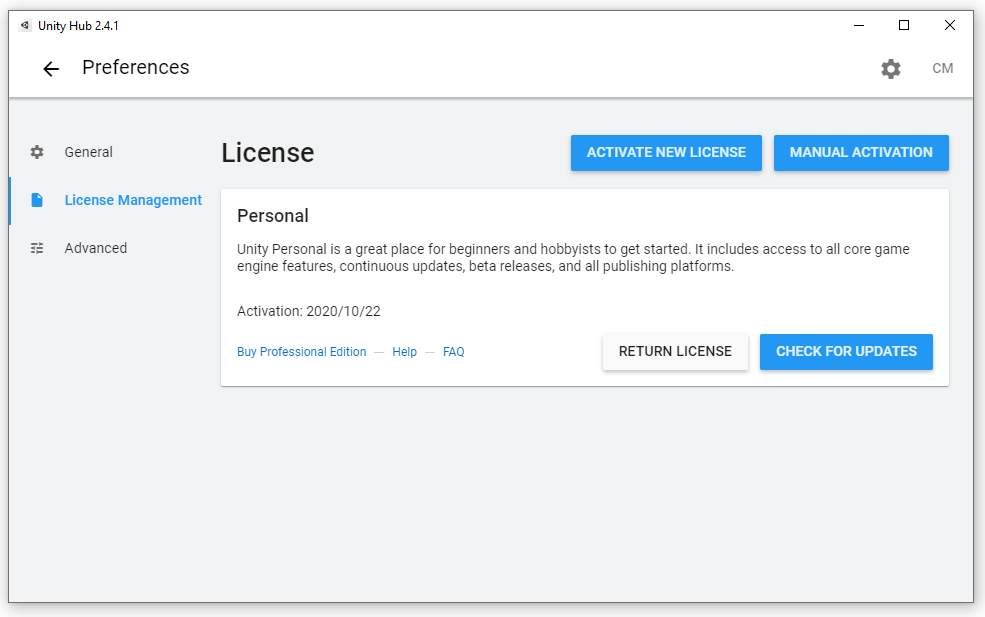


### **Creating a Unity Account**

1. If you already have a Unity account, click the icon in the upper right corner and then click sign in at the bottom of the new window as shown on the right. If you do not have an account, follow the steps above and when prompted to sign in, click the “create one” link as shown below.



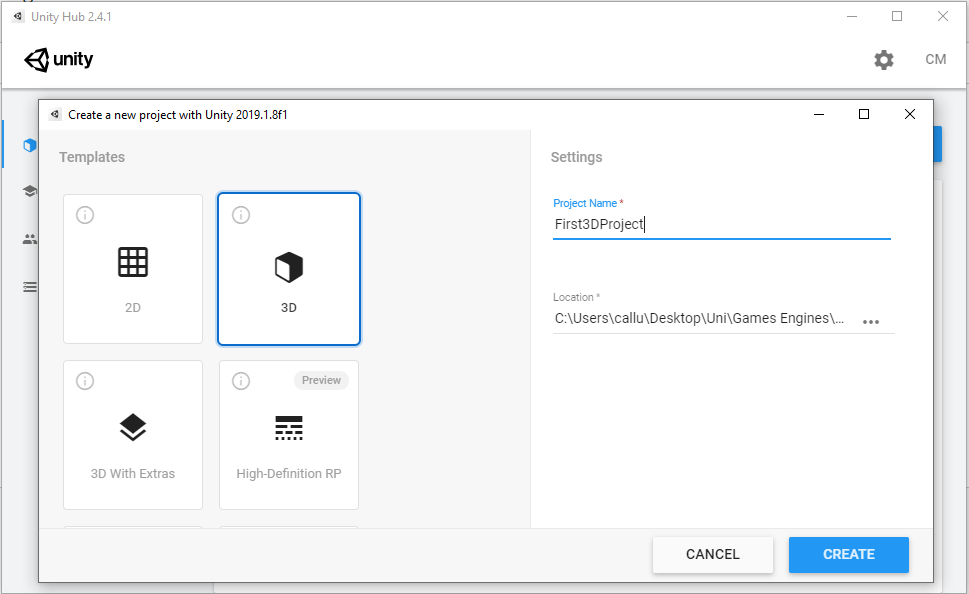
1. Fill in the account creation forms and click “Create Unity ID”. You can also sign up for a Unity Account using a google or Facebook account if you wish.
2. After completing and submitting the account creation forms, you must navigate to the specified email address you used to sign up for the account and click the confirmation link to complete the sign-up process.
3. After confirming your email address, go back to Unity Hub and click the account icon in the top right, it should now have your initials on it if you are signed in.
4. Click the manage licenses button and make sure you have a “Personal” license as shown below. If you do not, click the “Activate a new license” button and click “Unity Personal”, followed by “I don’t use Unity in a professional capacity” and click “Done”. You should then have a “Personal” License.



1. After you obtain the license, you are now ready to move onto the next part of the guide and start using Unity!

### **How to set up a Unity3D Project**

1. To create a new project, navigate to the “Projects” tab on the left and click the blue “New” button to be prompted with another window. Select the 3D option from the 4 options on the left, name the project and choose its storage location if necessary and click create as shown below.

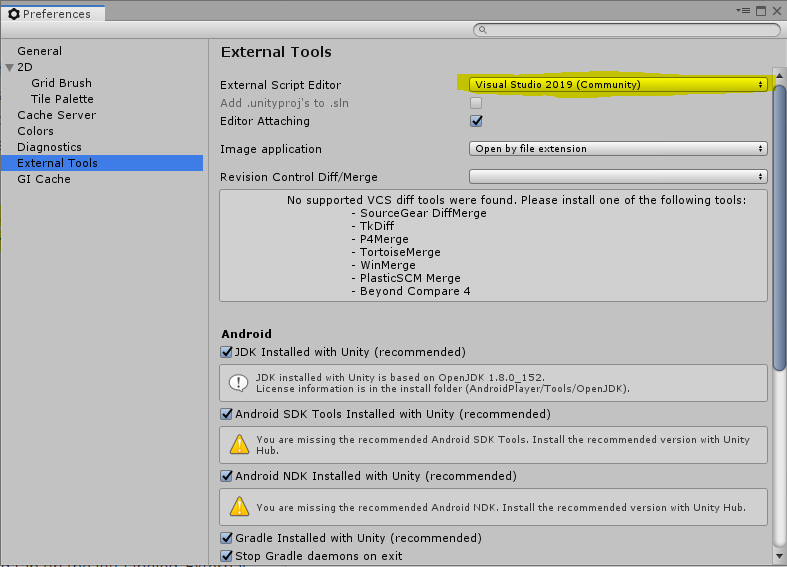


1. The first time you create a new project, Unity will take a bit longer to start-up. Once the start-up is completed and your project is created, you should see an interface like the one shown below.



### **Changing default editors**

Unity also has a preference setting that allows you to easily change the External Script Editor from the default. To change the editor, go to the top nav bar and select Edit > Preferences > External Tools. The top drop-down menu will allow you to change to an editor of your choice, my default is Visual Studio 2019 as shown below. Unity will detect any editors that you have downloaded if it does not you can choose the file path and that will add it as an option.

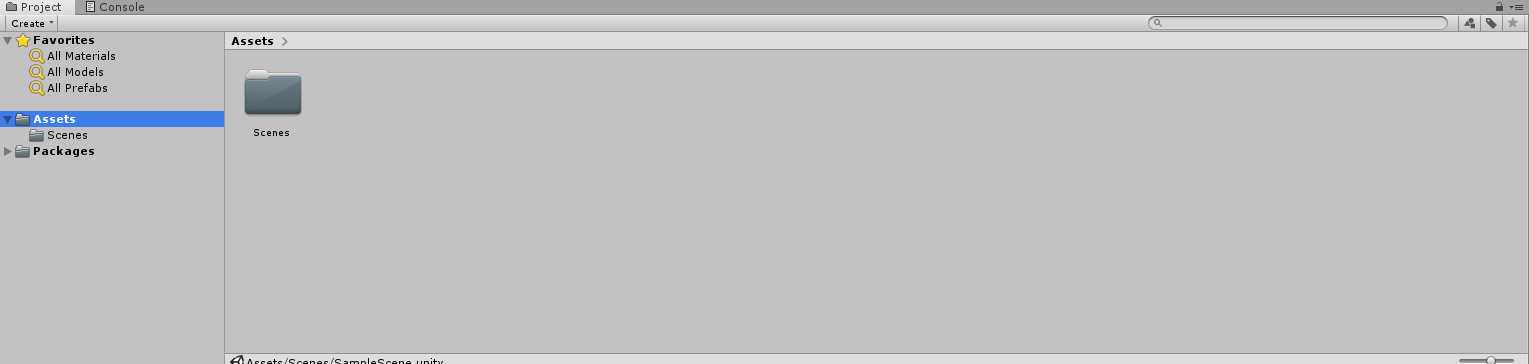


### **Layout Options**

There are several views that make up the editor window. You can move these views freely and place them in whatever position is most comfortable for you. For the purpose of this guide, I will keep the default layout as I give a brief description of each of the main views.

#### Project View

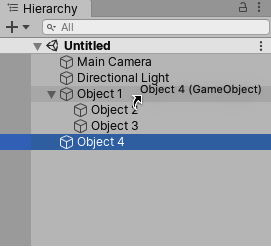
The project view is located at the bottom of the screen and shows the components that makeup the project and all of the assets attached to that project. An asset can be any file that is related to the project, such as scripts, textures, sounds, and meshes. You can also import assets from the assets store, for example, clicking on the “All Prefabs” option then selecting “Asset Store” will load all available assets on the store that you can load into your project.



#### Hierarchy View

The next view we will review is the hierarchy view. It shows all the components that are associated to a single scene rather than the entire project. A unity project will consist of multiple scenes that are linked together, each containing their own assets. A scene can be compared to a level in a game. A game may have multiple levels like unity projects have multiple scenes that makeup the game.

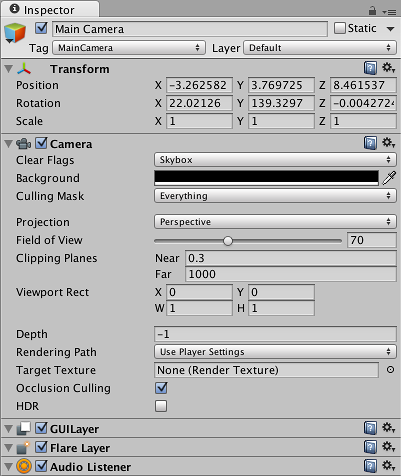
The hierarchy of the components that makeup the scene and how they are listed in the view can be seen below:



#### Inspector View

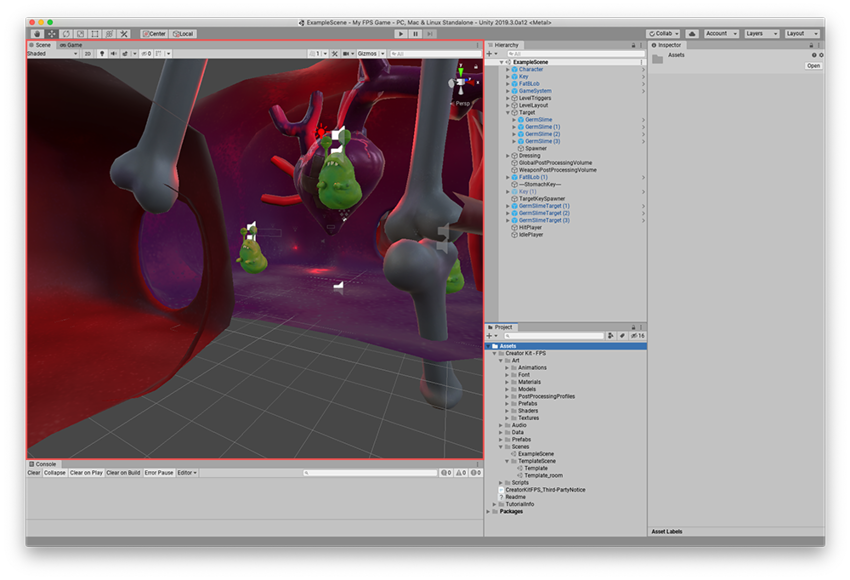
The inspector view allows the user to view and edit the properties and settings of most things in the Unity Editor, including the physical game components and items such as game objects, assets, and materials.

An example of the inspector view focusing on the main camera component of the game, can be seen below:



#### Scene View

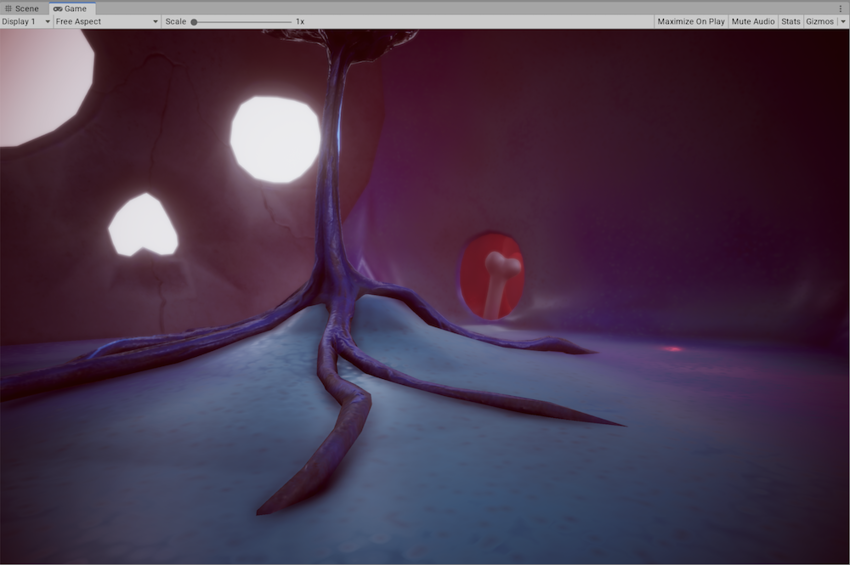
The scene view is an interactive view that shows you the world you are creating. You can use it to select and position scenery, characters, Cameras, lights, and other game objects. An example of the scene view can be seen below:



#### Game View

The game view is rendered from the camera in the project. It shows what your final application will look like with all effects and lighting playing out when the scene is viewed though this view and you need to use one or more cameras to control what the player sees when they’re playing the game.

Below is an example of the game view as well as the controls associated with it:



Game view Controls



Game view control bar



### Can I download an Asset without opening the Unity Editor? Is it possible to use Unity Assets within another game engine? – Unity**The Asset Store**

The asset store can be accessed through the unity website or the unity game engine itself. It is essentially a marketplace of assets that you can use and purchase assets to use in your game creation. You can also find an abundance of assets that are available to download and use free of charge. To find the asset store on the unity game engine application, open your project and go to **Window > Asset Store**. This will open a new tab in the editor and display the asset store.

# **Task 2 – Environment Design**

The second portfolio task was to create a 3D environment in unity using a consistent theme or genre. The focus of the task is to create a seamless level with a set list of required features such as terrain with hills, textures, vegetation, water, a skybox, and a lens flare. There are other features recommended to be added such as additional 3D models, lighting, and colour grading.

The set theme which was chosen was medieval and I wanted to build something like a blacksmiths house and forge. To start forming the terrain I used the “raise and lower terrain” option on the paint terrain tab. I formed a small pond to one side of the terrain and added a rock texture to the edges and bottom of the pond as well as some water. I also tried to get some foliage for under the water but could not find anything suitable on the assets store. Next Painted the terrain with the grass texture, added some trees, grass, and painted a pathway near where the blacksmith forge would be eventually.



Next, I placed some assets down like the house, forge, anvil windmill and cart etc. I was able to add colliders to most objects, but I had problems with others like one of the rocks. Whenever I added a collider to the last rock, it kept spawning it a way away from the rock itself and I could not position it properly. I read up on it online and seen a suggestion to add a cube, remove the rendering of the cube then add a collider to it as a child of the cube, but after trying it, it seems too complex for such a simple task and I gave up after a few tries. The article can be seen in the appendices under Ref.1.

After adding all the assets and placing everything as I wanted to, I starting adding the lens flare and succeeded into doing so only the lens flare shows up through solid objects that don’t have colliders and I didn’t have enough time to fix this for every asset, I was able to fix it for the buildings and other larger objects. After adding the lens flare, I researched how I could add some more detail or extra touches to the scene, and I came across a YouTube tutorial on adding fog and wind effects to the scene, (Jimmy Vegas, July 2017). I watched this tutorial and was able to add wind effects and fog to my scene which makes it a lot more believable. This video can be seen in the appendices under Ref.2.



The last few touches I added was adding the directional light and directing it appropriately as well as adding an fps counter to ensure the performance was solid when testing the level, making sure it had a consistent frame rate.

# **Task 3 – Character Controller**

Creating the character controller can be difficult at first and sometimes takes a lot of testing to get the values right for speed and jump height etc. A simple first-person controller was created which includes walking in any direction, sprinting (left shift) and mouse look. It also has a jump control (spacebar) complete with walking and jumping sounds.

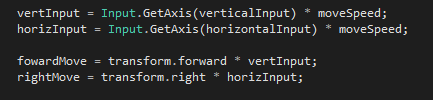
To start off a simple scene was created which would allow the controller to be tested. This was imported straight from the unity assets store. After creating an empty game object, a character controller was attached to the object as a child. The player camera was then shifted up by 0.9 units to give a realistic view and height suitable for a first-person view.

After creating the camera the playerLook, playerMove and footsteps scripts were created. Serialize Field was used to keep the variables private to their class but allow them to show on the editor. Variables were created to store the names of incoming input for the X and Y axis of the mouse as well as the sensitivity.

The input of the mouse was captured by assigning the “Mouse X” and “Mouse Y” in the editor after the variables were created. The input is taken from the mouse on each axis then multiplied by the mouse sensitivity and deltaTime. An axis Clamp variable was also created for the X axis so the user could only have a realistic X axis and not be able to turn the camera 360 degrees on the x axis as this is not realistic.

The xAxisClamp variable would lock the axis if it was greater than 90 degrees or less than -90 degrees, meaning the user can look up and down to a certain degree on the x axis but not rotate the x axis the whole way around and through what would be the players body. The camera is rotated by grabbing the cameras transform and calling the rotate function and passing in the amount you want the camera to rotate, which is -transform.right \* mouseY. The function at the end of the playerLook script was made to stop the camera from exceeding the clamp.

The player move script was essentially the same as the player look script in principle, keys were assigned, and I used:



to calculate how much the player should be moving with each key press.

The sprint function was created using MathF.Lerp which allowed me to multiply the base speed by the sprint speed if the shift key was actively being pressed. The jump method was also implemented using Vector3.up and multiplying it by the jump height and jump multiplier as well as delta time whenever the function is called. The function is called whenever the jump input function is called which is the bound to the spacebar.

The footsteps script was simple to create and uses an update method to determine whether or not the player is on the ground and moving, if both conditions are true and the audio is not playing, it then plays the footsteps audio. A similar method was used for the jump sound although I could not get it to work properly and differentiate between the two audio sources, therefore both the walk and jump sound are the same sound, the walking sound.

# **Task 4 – Collision Detection**

To test the collision detection a jump puzzle was created using some assets from the unity store. The character controller from Task 3 is also implemented into the level. The collision messages show on screen when the player hits the main floor at the start of the level, when they hit on of the 4 main walls and also when they reach the end of the level at the pink platform at the top of the room.

The messages were implemented using triggers, the OnTriggerEnter and OnTriggerExit methods. Instead of using the main walls and the colliders attached to them to activate these methods, new cube objects were created and then the rendering mesh was removed from each of them to create invisible triggers. The reason this was done was because the previous attempt involved using the colliders on the main walls and floors but this lead to the player falling through the floor after the text had changed and a fix couldn’t be found for it so a new approach was chosen.

When the player enters the “invisible boxes” on the walls and floors, the OnTriggerEnter method id activated and the message “You have hit a wall!”, is shown on the screen, when the player then leaves the bounds of the wall/invisible box, the message is cleared from the screen. A message is also shown when the player hits the floor and is removed thereafter when they leave the bounds of the floor and when they enter the bounds of the final platform to say that they have completed the level.

To display the text on screen, a canvas and text object were used and placed at the bottom of the screen. A text object was then declared in each script so the text object could be dragged onto the property slot in the editor. An example of the text that is shown on the game screen can be seen below:



# **Appendices**

## Assets Used (All from the Unity asset store)

1. Unity Standards Assets Pack by Unity
2. SpeedTutor – Tutorial Scene – FREE by Speedtutor
3. AllSkyFree by RPGWhiteLock
4. Grass and Flowers Pack 1 by Vladislav Pochezhertsev
5. Medieval Village by Asset Maiden
6. Tree9 by Rakshi games

## Research

### Ref. 1

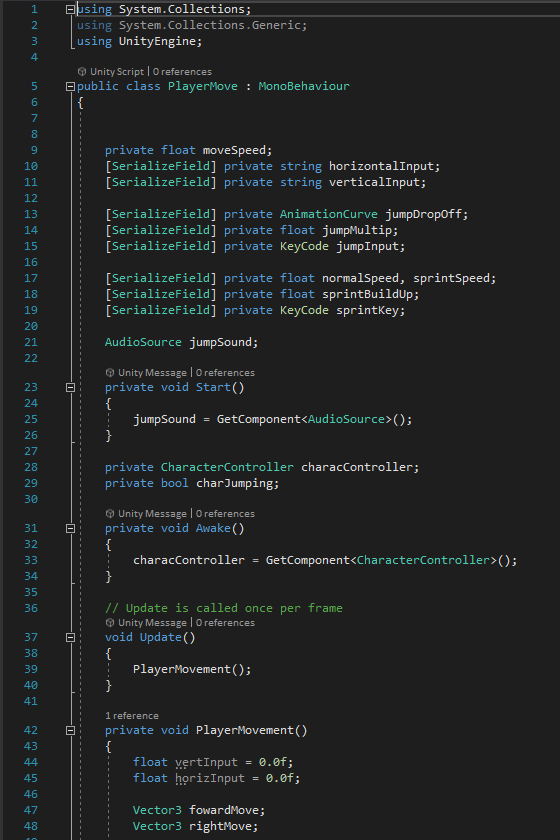
Rotate Collider Box Unity Forums Posted by “Delta12” 22nd March 2014, https://forum.unity.com/threads/rotate-box-collider.3186/

### Ref. 2

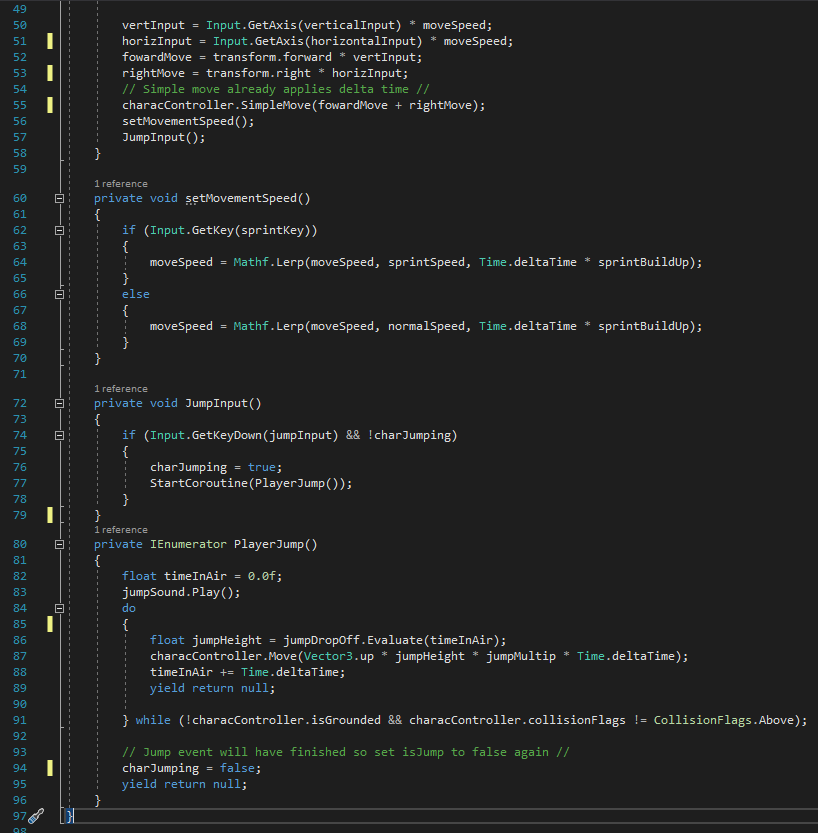
Mini Unity Tutorial - How to Create Fog & Night Illusion – Uploaded by Jimmy Vegas, 1st July 2017, <https://www.youtube.com/watch?v=UgJE3TgT3o8&feature=emb_logo>

## Scripts

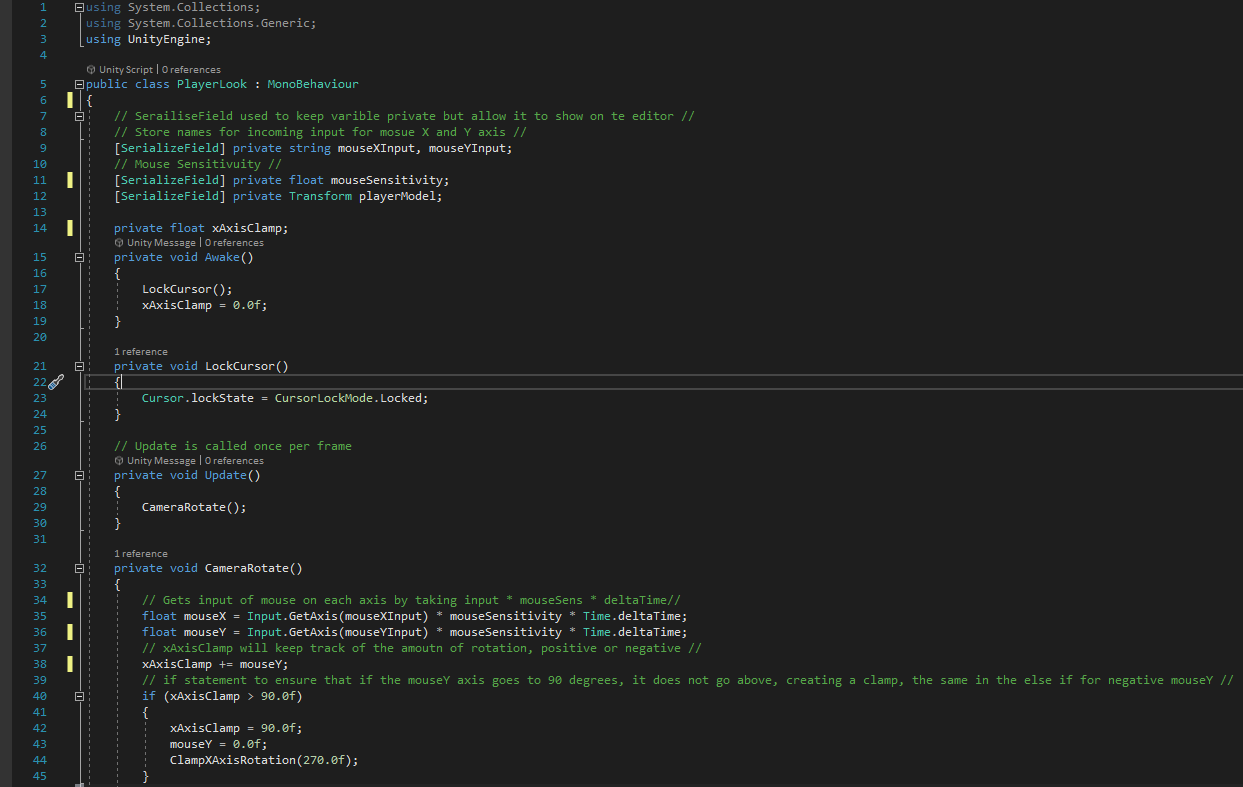
## PlayerMove (1)

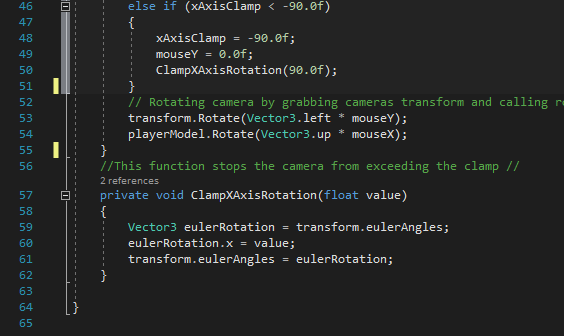


## PlayerMove (2)

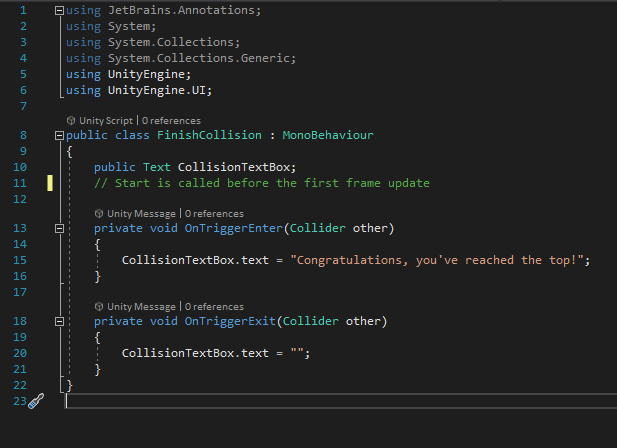


## PlayerLook

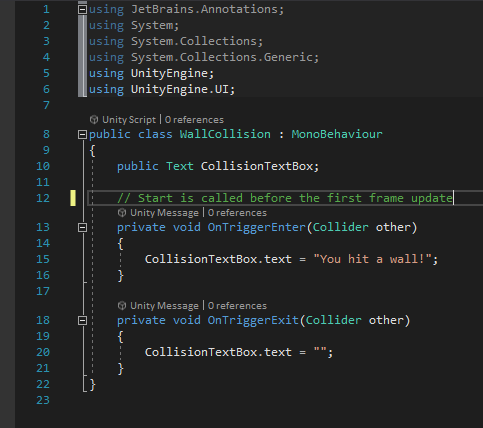




## FinishCollision



## WallCollision



## FloorCollision

