UNITY GUIDE

INDEX:

1. Brief Introduction to Unity and Editor
   1. Unity Windows & Layout
   2. Hierarchy & Inspector
   3. Project Files
   4. Other Notes
2. Brief Introduction to C# in Unity Game Engine.
   1. Script Layout and Comments
   2. Declaring, Displaying & Editing Variables
   3. Unity Functions
3. Video Tutorials

Because we are using Unity as our game engine, and I am aware that the first years have not had much experience with it, I have decided to put together this guide that will hopefully get you up and running in Unity. This short guide is an overview of what Unity is, and more in-depth tutorials will be available at the end of the document, as well as some programming tips and some ways that I write my code, but because this is a group project, if you have any questions or advices make sure to tell me or Jamie and we will try to help you out best to our experience. – Daniel.

Brief Introduction to Unity and the Editor.

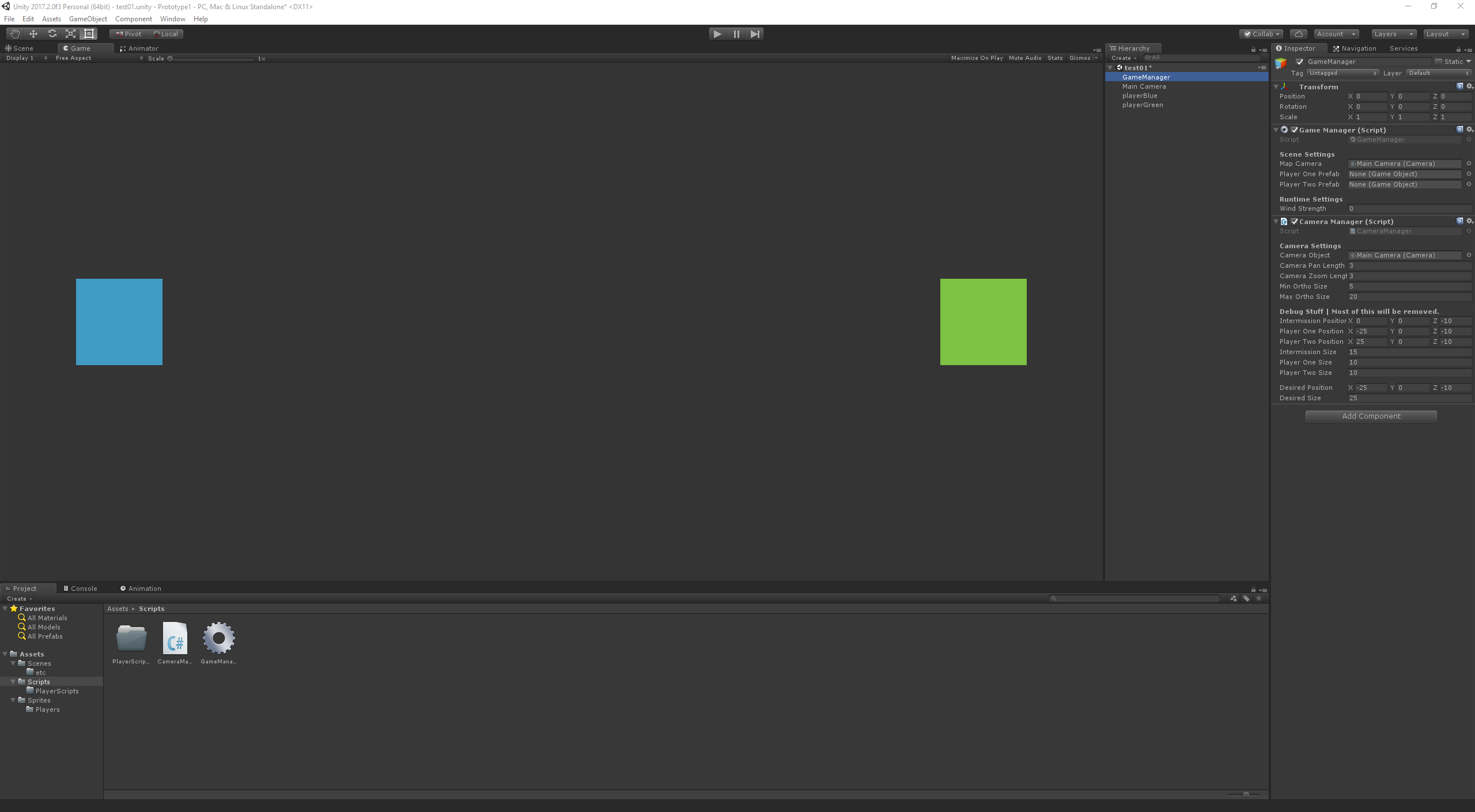
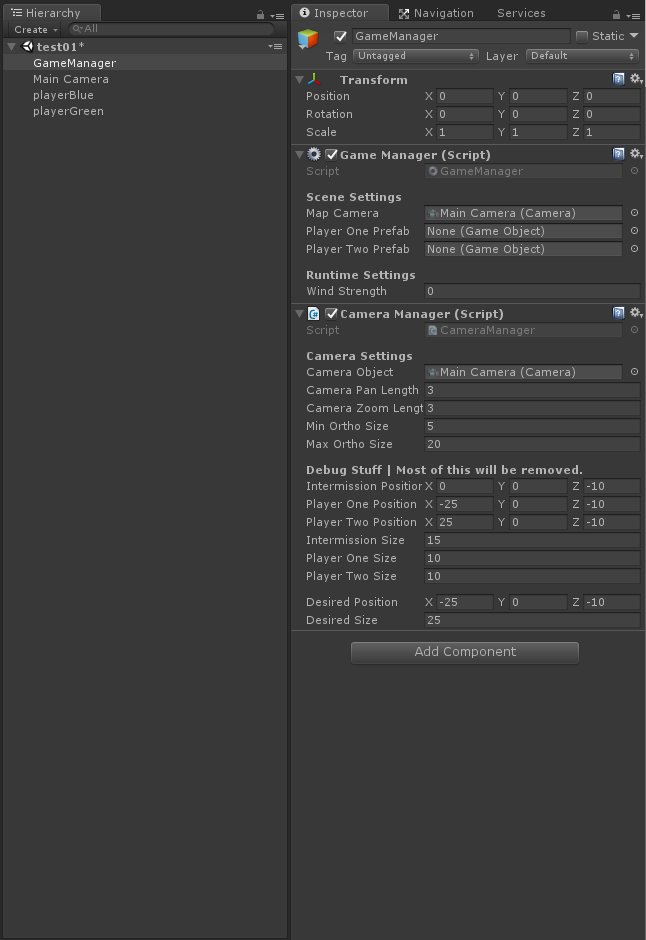


Figure 1: My custom Unity window layout.

*Unity windows & layout*

When you first open Unity, you are greeted with the default layout of the windows, the beautiful thing about Unity is that is so modular, meaning you can drag all the windows around; disable any that you don’t want to see, or you can even create your own ones! Make sure to set it up in a way that is most comfortable for you to use, and if you have a multi-monitor setup, you can take advantage of that and have scene view on one monitor, whilst the game on another (allows you to see what is happening better).

As you can see from the screenshot I have moved my windows around from the default view so that it suits my working space, I tend to move the windows around depending on what I do but this is generally layout that I use. If you find your favorite layout, you can save it by clicking on “Layout” in top right and pressing “Save Layout…”; this way you can always go back to this same layout.

*Hierarchy and Inspector windows*

Hierarchy and Inspector windows are the two most important windows in the editor when you are working on project;

*Hierarchy* is the window that shows all objects that are currently present in the scene, this includes any empty objects, lights, etc. Here you can easily find a desired game object that is currently somewhere in the scene. When an object is inside another, we call it the ‘child’ of the object; for example, is playerGreen is inside the playerBlue object, we would say that playerGreen is a child object of playerBlue and playerBlue is the parent of playerGreen. This can seem confusing at first, but you will soon understand it better; it is good to use the proper jargon, to make it easier for everyone to understand the idea.

*Inspector* window is where all the settings for the currently selected game object will appear, so for example on the left I have selected the ‘GameManager’, in the Inspector I can see all the settings I have created in the script; but not only that, I can also other settings such as the ‘Transform’ (position, size, rotation) of the object; if the object had a 3D mesh, I would be also able to change the mesh in this window. The “Navigation” and “Services” tab you can ignore for now, as neither of them will be used in our group project.

Figure 2: Hierarchy and Inspector tabs in Unity Engine.

*Organizing the Project (folder framework)*

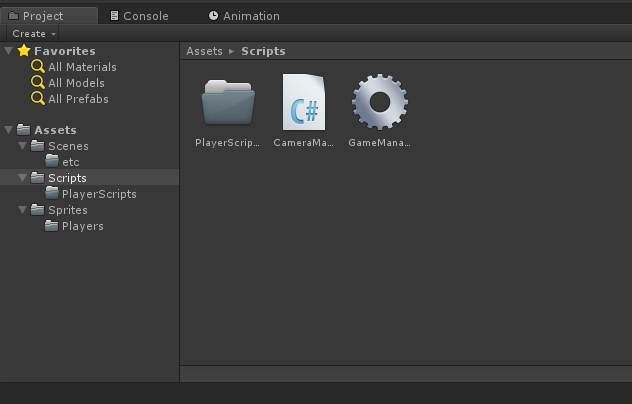
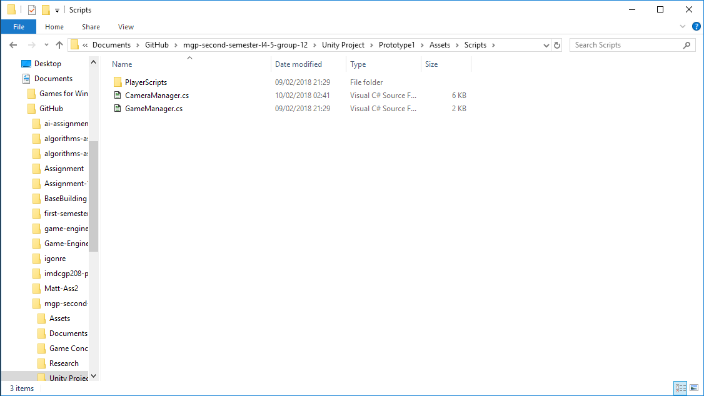


Figure 3: Unity Folders and Files in Explorer (left), and same files and folders as seen in Unity Editor (right).

Creating folders and files in Unity is very simple, all you need to do is right click in the project tab and select the item you would like to create. Folders can be created in both the editor or using Explorer, but its better to do it in the editor, as every time you add anything outside of the editor Unity will need to import that item; although this is avoided when creating folders in the editor.

For the project I have created a general layout of the folders, so that we can keep it neat and easy to navigate around; this way any file we need, we can easily find. Generally, as a rule of thumb we should stick to the same folder and file naming, to avoid confusion (remember the project repo is public, this means that anyone can view it; so, keep it neat and organised).

*Other Notes*

This should be all the information you will need to get started with Unity, below I have left some links to Unity Tutorial projects which you can import into your project and tweak around and get used to the game engine. You can also tweak things around in the Group Project, and read through the scripts and get some basic understanding of them.

You can also look through the other resources down below to learn furher more about Unity, and I also recommend watching some of the tutorials if you are still finding it difficult to get used to the Unity engine.

In the next paragraph I will quickly describe the way I will write the C# scripts in our group projects, and overall how C# works in Unity, to give you better understanding of programming in Unity.

Brief Introduction to C# in Unity Game Engine.

In this part I will briefly talk about using C# in Unity Engine, I won’t go into deep coding as that would take too much time, and you are currently being taught C# as we go; thus, I’d like to see your progress in the group project. When it comes to designers, you can also come to the Unity C# programming lessons, where you can learn some basic scripting; this is highly recommended, because you will be able to understand the code better and it’ll make working with programmers easier.

*Script Layout and Comments*

In all the scripts that I will work on for the group project, I will keep the same layout by creating a short introduction in the script to say what exactly it is doing and how it is doing it; this way you can easily jump into it and understand what is happening in the script.

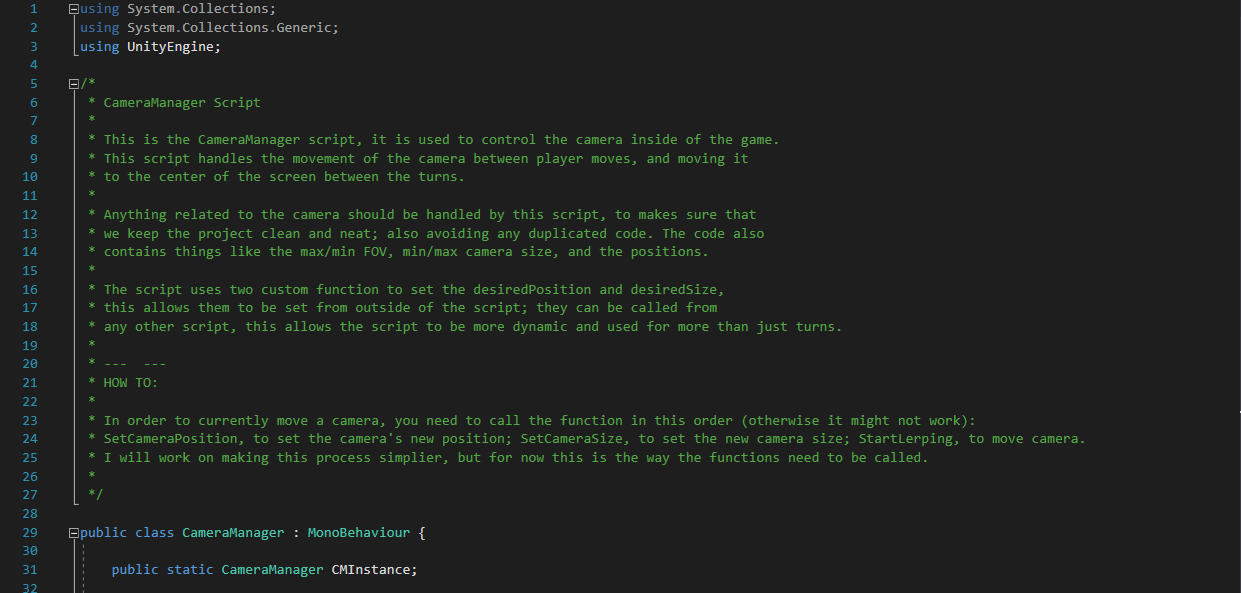


Figure 4: Image from the CameraManager.cs, showing introduction comment.

As visible in the above screenshot, the few paragraphs are telling the reader how the script works and what it should be doing; if you have any questions about any of the scripts in the project, make sure to let me know otherwise I will assume that you are understanding the scripts and you don’t need any help.

At the top of each script are the libraries that the script is currently using, by default when creating a script in Unity, it will add all the generic libraries to the script to make your life easier (you can change that in the project settings), but you can just delete those lines and you will no longer have access to those libraries. It is important to keep the UnityEngine library, otherwise you will not have access to any of the Unity functions.

Next below that is the comment that I will be leaving on each script, as mentioned above. Below that is where we declare the namespace and variables for the script. Currently you don’t need to know about Namespaces, but if we will use them I will make sure to explain them to you (unless you are interested, then let me know). After declaring the variables, we declare the functions; we can either use Unity built in functions or we can create our own functions; I will explain those in class as it is easier to do while in person.

*Declaring, Displaying & Editing Variables*

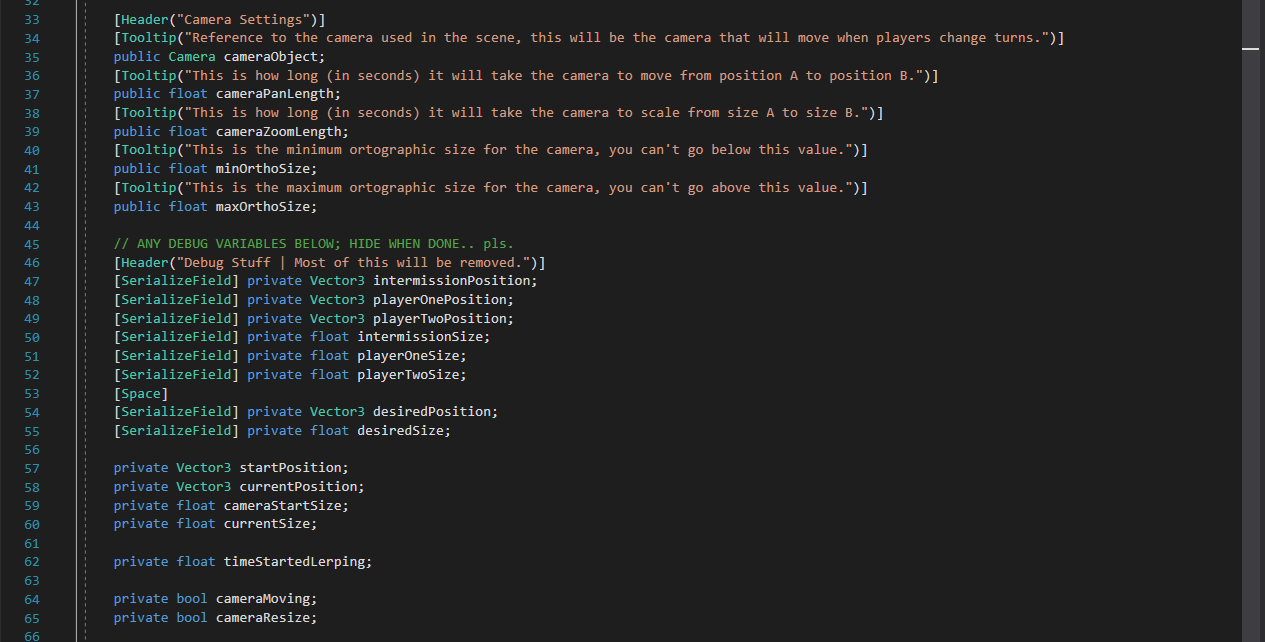


Figure 5: Public and Private variables that are declared in the CameraManager.cs script.

When working on our project we will mostly use either Public or Private variables; using the above as example, when I create the cameraObject variable I make it public, this means that by default the variable can be accessed and edited in the inspector (as seen on the screenshot on the right); this way I’m making sure that anyone working on this script can easily edit the variable.

Figure 6: Variable as seen in Unity Editor.

From programming side, making variable public means that other scripts can access it and change the value, for example I could create another script that replaces the “CameraObject” with another camera if there are multiple cameras in the scene; for example imagine game like Resident Evil (old PS1), where the camera would change depending on where the character is standing; this same way I could create a script that tells my manager which camera should be currently active and I replace it using the script.

Now in some cases we don’t want certain variables to be visible, for example when we have currency in our game we don’t want it to be able to be changed by any odd script, as this could easily lead to errors and weird currency results (game breaking bugs possible); thus we would make the currency private, this means only the script where the variable is declared can change the values of the variable; in that case we could have some sort of CurrencyManager to add or remove currency from the player.

*Unity Functions*

In Unity, by default, we are given some functions to work with; and you could even build a whole game using just Unity functions; but creating your own functions gives you better control of what the function can and can’t do.

By default when we create a script we have two functions, “Start ( )” and “Update ( )”; Start is called whenever the script has been created and the game is launched; this is where you can assign variables on the script. For example, if you had a player who had experience saved into a XML file, this is where you would read and assign the experience points back whenever loading the game.

Update function is run every frame, this means that any code you have in here will be run every game frame. Be careful, because it is easy to crash Unity if run into an infinite loop, or something that clogs up your RAM (I’ve been there myself); so, make sure to proof read your code before you do anything complex.

Another function that we have is “Awake ( )”, and this is run whenever the script is first created no matter if other objects have loaded into the scene; so if you are assigning an object to a variable here, it might not work because the function might be called before the other object has even been created.

“OnCollisionEnter ( )”, “OnCollisionStay ( )” and “OnCollisionExit ( )” are Unity’s built in collision detection functions; they will return whatever has collided with the current game object (assuming both of the objects have a collider on them); instead using Collision we can use Trigger, this means that the object has to be inside the Collider (tick the “Trigger” option on the collider), and this is used to detect if player has enter the collision zone; for example this can be used to trigger events, such as a cutscene.

Unity has loads and loads of functions available to use, here I have described the most used ones and the ones you will see the most in the project. For more information I recommend you looking into Unity’s API which I have linked down blow.

Video Tutorials

[](https://www.youtube.com/watch?v=Ep0rlBQRcVc)

*MisterNinjaBoy – Unity 5 Tutorial: Basics*

This is a good tutorial to watch to get started with Unity, he goes over the basics of using Unity as a game engine; he does not go into much coding, mostly goes over the Editor and what I have described in the document.

[](https://www.youtube.com/watch?v=qwuPiaFU37w)*GameGrind – Making a Simple Game in Unity*

This tutorial goes into C# programming, it introduces you to simple Unity coding and will guide to make a very simple, but fun game.

Although the tutorial goes into programming, it is still a good watch because he goes over many things that are not just programming related and goes over the basic code in detail, so it is very good watch to get the basics of using C#.

[](https://www.youtube.com/watch?v=IlKaB1etrik)

*Brackeys – Making a Basic Game*

Brackeys is a channel that delivers very good quality videos, that will guide to use Unity. I personally use it to learn new things every day and he goes into detail when explaining what he is currently doing on the screen.

Once again this is a programming tutorial that shows how to create a basic game, but once again it is a very interesting watch as he goes over many details of using Unity Engine and not just purely the code.

I recommend watching the whole series, as they will get you right into Unity.

[](https://www.youtube.com/watch?v=UbPiCgCkHTE)*Brackeys – Making a 2D Platformer*

Once again this is a tutorial series by Brackeys, but this time he is showing how to create a simple 2D platformer in Unity; this is somewhat similar to what we are working on currently, and would be a great starting point to get grasp when it comes to programming.

I would also recommend subscribing to his channel, because he uploads a lot of good content, sometimes programming related and sometimes graphics like Particle Effects, Shaders and etc.