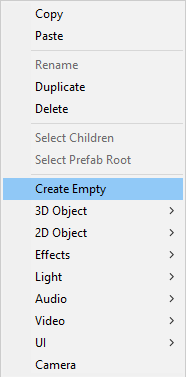
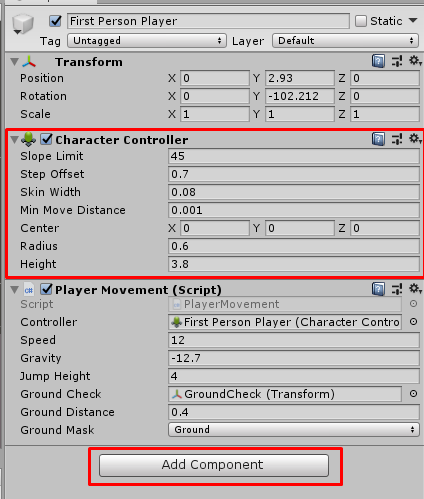
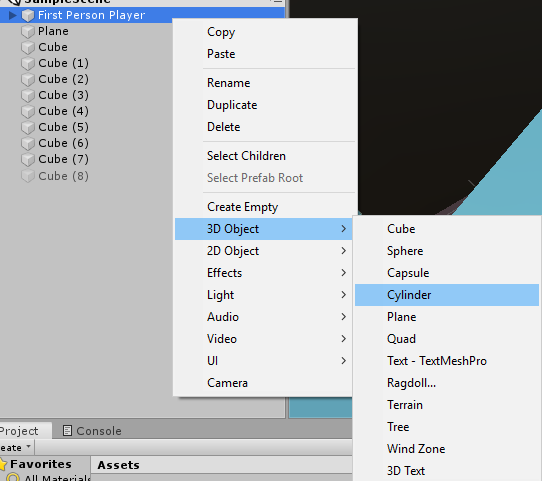
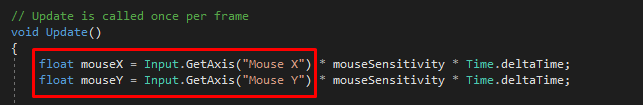
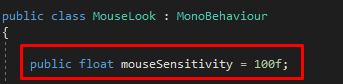
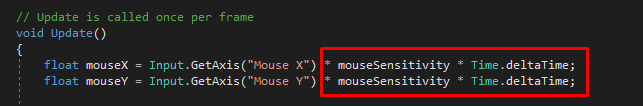
**Tutorial 1 – First Person Camera**

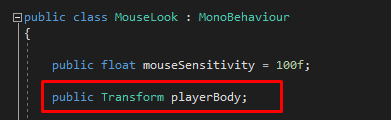
* The first thing to do is to create an empty object in the Hierarchy and rename it to “First Person Controller”. Place it onto the ground of your scene (make sure it has a collider).

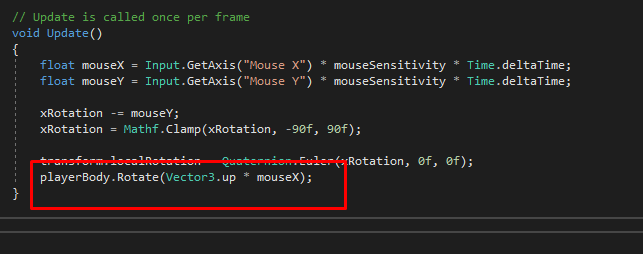


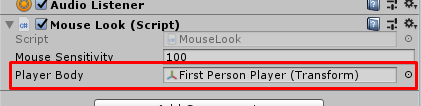
* You then want to add the component “Character Controller” onto the object you just created.
* To make your player visible to you, you want to add an object to the Player to make it easier to identify. You do that by pressing the right click onto the First Person Player. Once created, remove the capsule collider from the “Character controller, as the Cylinder already acts as its own collider.
* Now drag the “Main Camera” from the Hierarchy into your First Person Player component, or if you don’t have one, simply create a new one by right-clicking the component and adding a camera. Reset the transform after its been added to the component.
* Make sure you move the camera to the top of your cylinder to create a more realistic First-Person experience and avoid collision problems later on, as well as keeping it central, inside the cylinder object, to be able to see through it.

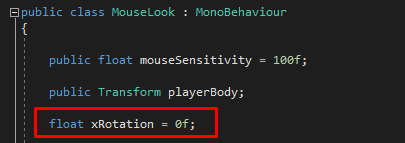
**Adding the script onto the camera**

* Select the camera, and add a script component and call it “MouseLook”. Then double-click it and open it in your preferred programming software.
* To gather the input from your mouse, you need to add an X and Y axis float in your void Update section.
* If you want to be able to change the sensitivity in Unity itself, you have to create a “public float” for the sensitivity in the “public class” section and set it do a default speed, which you then can later edit to your liking.
* You want to make sure to update your mouse input script to catch onto your set mouse sensitivity, as well as make sure for it to update independent of the current framerate, since the script is made in the update function, simply by multiplying it by “time.deltaTime”.
* To make sure the player rotates around the X axis, and not just the camera, you have to reference the camera with the entire First Person Player component, which you can simply do by creating a “public Transform” and calling it something like “playerBody”.



* We can now access the player’s body in the update method and make it rotate with the camera on a chosen axis.
* After saving the script, you can now link up the First Person Player component with the “mouseLook” script, by dragging it into the Player Body line.



* Now we want to add the possibility of moving around the Y axis. Start by creating a float in the public class section.
* Now in the void update section we make sure the xRotation uses the mouse’s Y axis’ movement, by also using the “Quaternion.Euler” so we can also clamp the movement of the Y axis, to avoid a possible full spin on the Y axis. To finish it off, we want to make sure the cursor is hidden once the game starts, so it does not just leave the window and click something else which it is not supposed to at the time, by simply locking the cursor.

