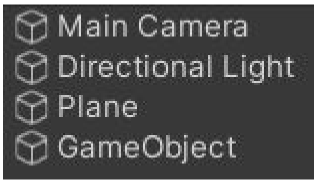
Tutorial 1: First Person Camera Movement

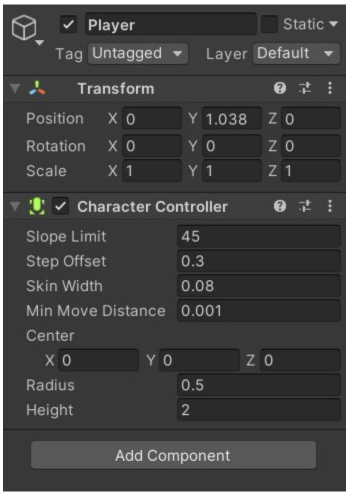
Preparation:

Create a grounding, you can use a Plane, and an empty GameObject that will be used as a container for the player.



Main Camera
Directional Light
Plane
GameObject

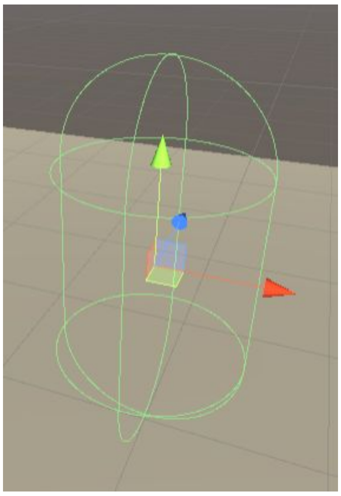
Inside the empty GameObject, that I've renamed as Player, add a Character Controller and you will be able to see a green skeleton for the Player.



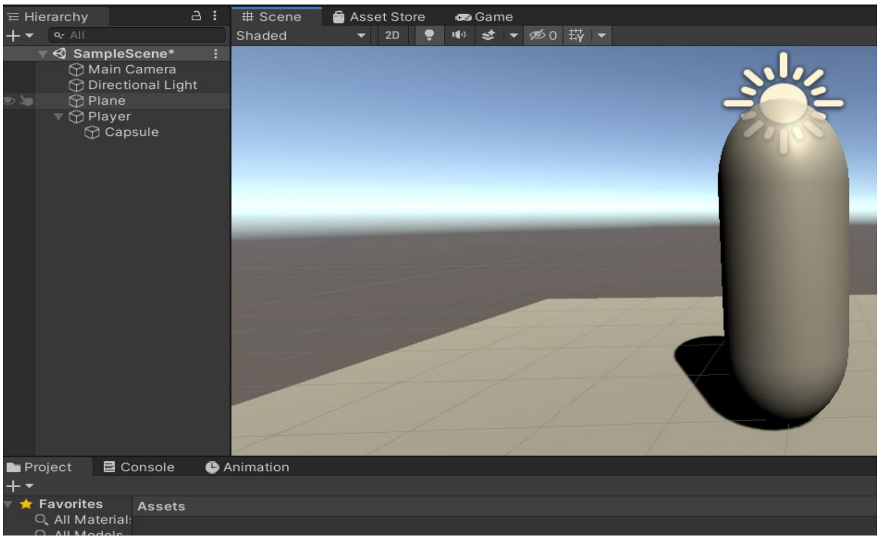
☑ Player ☐ Static
Tag Untagged Layer Default

Transform
Position X 0 Y 1.038 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

☑ Character Controller
Slope Limit 45
Step Offset 0.3
Skin Width 0.08
Min Move Distance 0.001
Center
X 0 Y 0 Z 0
Radius 0.5
Height 2

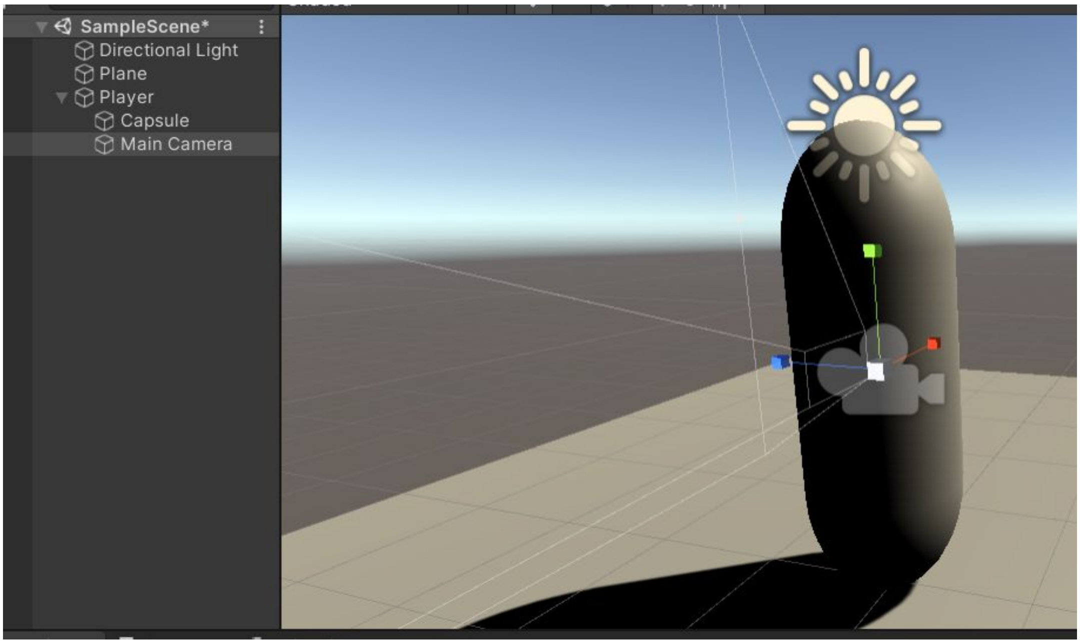


Change the radius or the height of the Controller if necessary and then add a 3D object inside the Player to make it easier to see it. I've added a capsule.

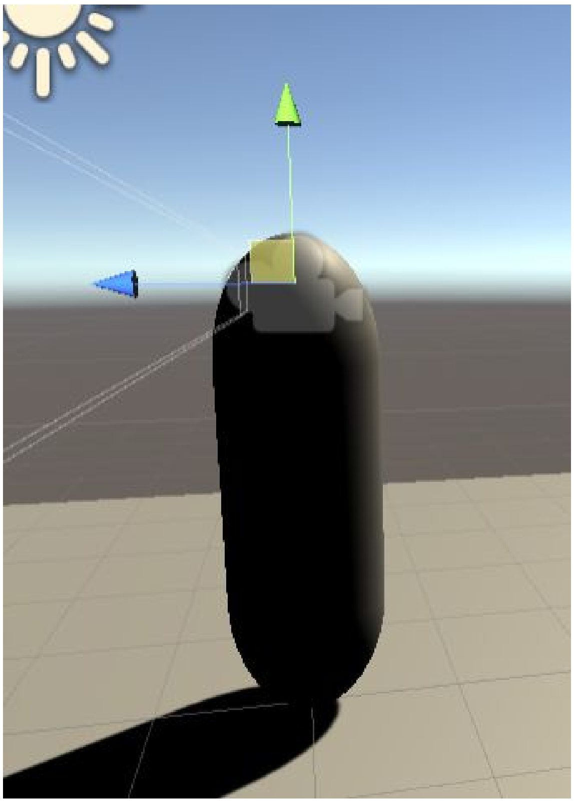


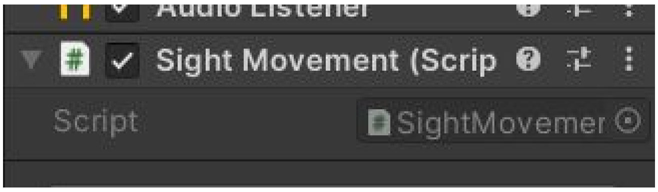
SampleScene*
Main Camera
Directional Light
Plane
Player
Capsule

Move the camera also inside the Player and reset it’s position.



Move the camera up to the position of the “head” of the Player and create a script for the camera.




Audio Listener

☑ Sight Movement (Scrip

Script SightMovemer

Code:

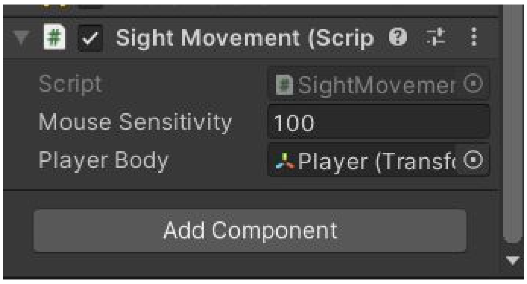
At the beginning of the code you will need to set some variables.
First, to control the sensitivity of the mouse movement:

```
public float mouseSensitivity = 200f;
```

You will need to create a relation to the Player from the script, for that use:

```
public Transform playerBody;
```

And from the Unity Inspector, add the Player GameObject to the camera script.



Sight Movement (Scrip

Script: SightMovemer

Mouse Sensitivity: 100

Player Body: Player (Transf

Now, create a variable for the vertical rotation of the camera.

```
private float xRotation = 0f;
```

Inside the Start function, write the next code to block the movement of the cursor to avoid clicking outside the game.

```
Cursor.lockState = CursorLockMode.Locked;
```

Inside the Update function, create two variables, one for the horizontal mouse input and the other for the vertical one.

```
float mouseX = Input.GetAxis("Mouse X") * mouseSensitivity * Time.deltaTime;
float mouseY = Input.GetAxis("Mouse Y") * mouseSensitivity * Time.deltaTime;
```

The vertical mouse input must rotate vertically the camera. So the variable xRotation needs to be related to the variable mouseY. To avoid the controls to be inverted, write the code like this:

```
xRotation -= mouseY;
```

Next, you need to avoid the camera to rotate 360º vertically to make the control easier and more realistic. To delimitate the angle the camera can rotate, write the next:

```
xRotation = Mathf.Clamp(xRotation, -90f, 90f);
```

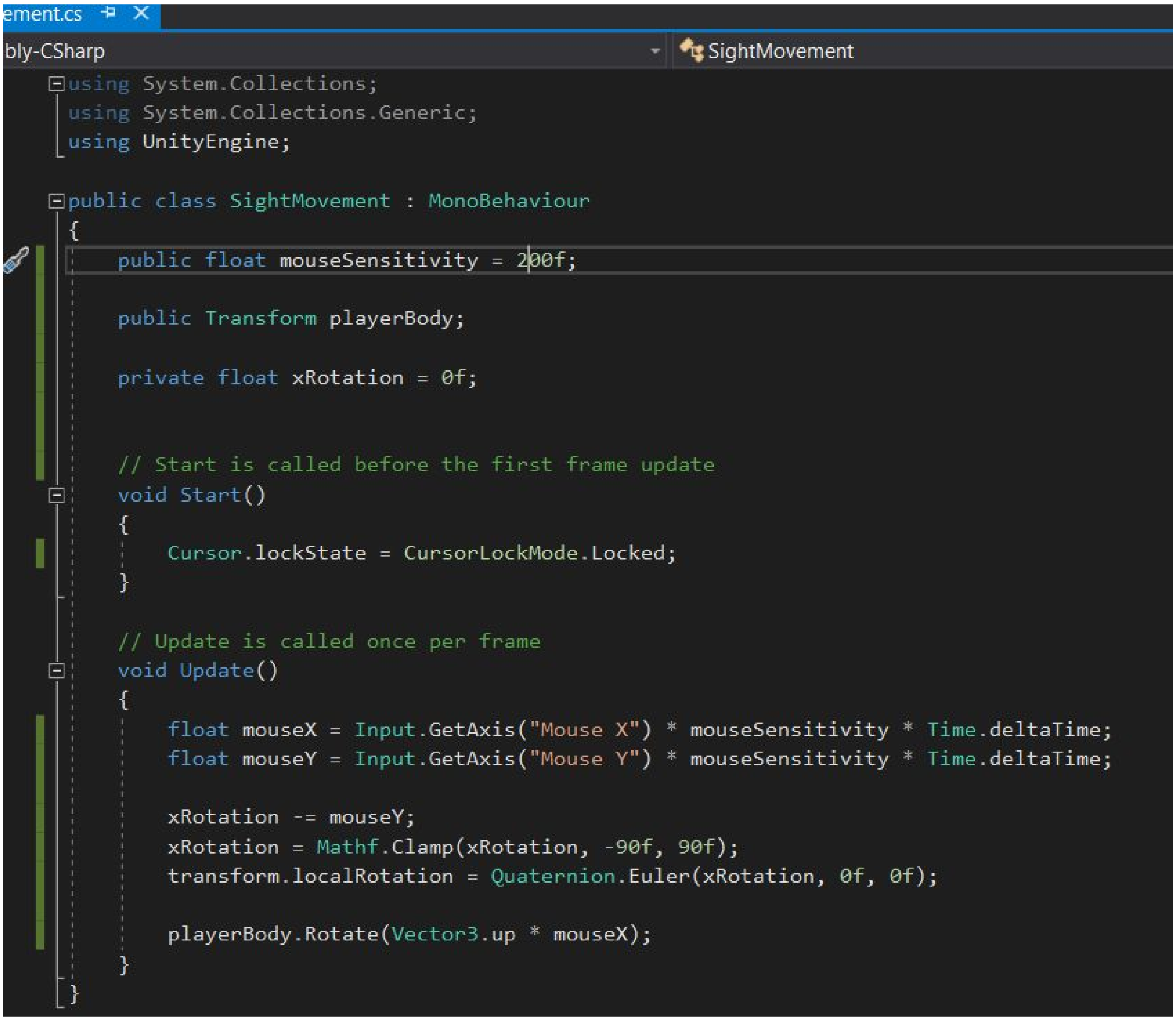
For the camera to realize the vertical rotation:

transform.localRotation = Quaternion.Euler(xRotation, 0f, 0f);

The only thing that is left is the code to turn the player from side to side, according to the horizontal mouse input.

playerBody.Rotate(Vector3.up * mouseX);

The script should look like this:



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

public class SightMovement : MonoBehaviour
{
    public float mouseSensitivity = 200f;

    public Transform playerBody;

    private float xRotation = 0f;


    // Start is called before the first frame update
    void Start()
    {
        Cursor.lockState = CursorLockMode.Locked;
    }

    // Update is called once per frame
    void Update()
    {
        float mouseX = Input.GetAxis("Mouse X") * mouseSensitivity * Time.deltaTime;
        float mouseY = Input.GetAxis("Mouse Y") * mouseSensitivity * Time.deltaTime;

        xRotation -= mouseY;
        xRotation = Mathf.Clamp(xRotation, -90f, 90f);
        transform.localRotation = Quaternion.Euler(xRotation, 0f, 0f);

        playerBody.Rotate(Vector3.up * mouseX);
    }
}
```

This tutorial is based in the one from the next link:
https://www.youtube.com/watch?v=_QajrabyTJc&t=1s&ab_channel=Brackeys