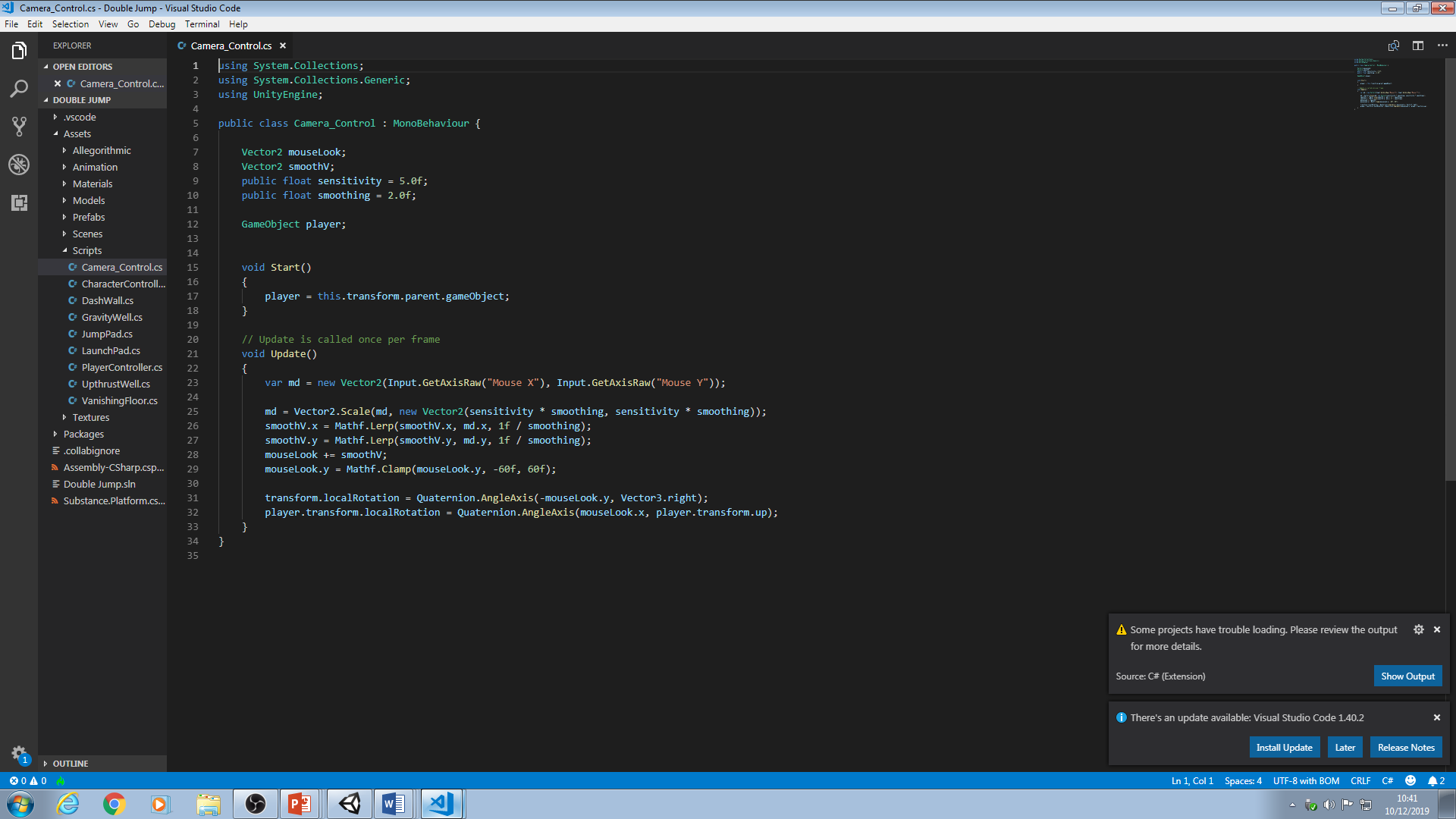
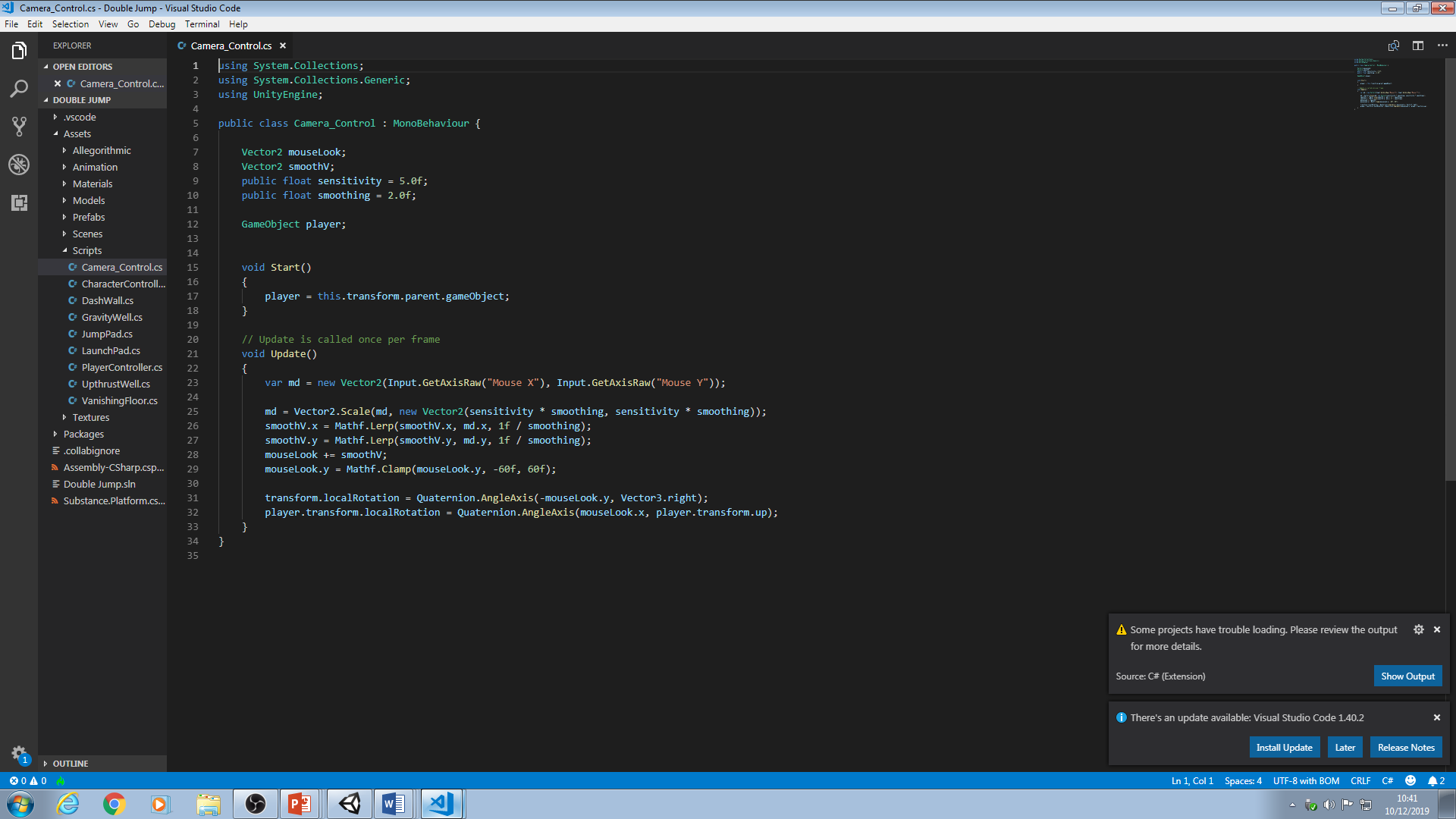
Tutorial 2: Camera Controller

For this tutorial I will show you how to code and make a Camera Controller for your character

1. First, move the Main Camera to the Player object as a child.
2. Create the variables as shown in the picture.
3. Next, in the Start method type player = this.transform.parent.gameObject; which will reference the parent object which is the Player
4. Next, in the Update method type var md = new Vector2(Input.GetAxisRaw(“Mouse X”), Input.GetAxisRaw(“Mouse Y”)); this gets the X and Y value of the mouse
5. Next, type md = Vector2.Scale(md, new Vector2(sensitivity \* smoothing, sensitivity \* smoothing)); Vector2.Scale multiplies 2 vectors together so the original md mouse values are multiplied with a new Vector2 that is made up of our smoothing and sensitivity values.
6. Next, type smoothV.x = Mathf.Lerp(smooth.x, md.x ,1f / smoothing); this sets the X value of the smoothV vector. Mathf.Lerp interpolates an object between two points by a set value – smooth.x, md.x being the interpolation points and 1f / smoothing being the value by which they interpolate
7. Next, copy and paste that line of code but change all the X values to Y so it becomes smoothV.y = Mathf.Lerp(smoothV.y, md.y, 1f / smoothing);
8. Next, type mouseLook += smoothV; this continually applies your adjusted mouse values to the new mouseLook vector every frame.
9. Mouselook.y = Mathf.Clamp(mouselook.y, -60f, 60f); this clamps the Y value of mouseLook so that the players Y vision can only go up and down so far.
10. Next, type transform.localRotation = Quaternion.AngleAxis(-mouseLook.y, Vector3.right); this transforms the camera around the Vector3.right axis based mouseLook value which moves the camera up and down
11. Finally, type player.transform.localRotation = Quaternion.AngleAxis(mouseLook.x, player.tranform.up); this transforms the player around an axis based on the x value so the camera moves left and right.