**Task/Tutorial Journal 1**

**16/10/2018:**

The initial jump script was not working upon being implemented onto the capsule. After checking the code, I couldn’t find any feasible fault in what could be preventing it from jumping at a first glance. Because of this, I decided to play the scene and figure out if anything peculiar was happening. I had discovered that the raycast wasn’t long enough to reach the surface that the capsule was sitting on, meaning the “onGround” variable was constantly being set to false, preventing the player from jumping. To fix this I simply extended the raycast length by an extra .5 from its initial 1.0 length.

Another issue that was apparent to me after implementing the “canDoubleJump” variable was that, when pressed quickly enough, the player could jump 3 times instead of the indicated 2. I immediately realised that this was more than likely to do with the raycast once more. As expected, the raycast was actually too long in length this time around, as the player was able to exploit quickly pressing the space key in order to have the “onGround” variable still set to “true” on the second jump, allowing for an additional third jump. The length was decreased from 1.5 to 1.3 to prevent this exploit.

**Task/Tutorial Journal 2**

**17/10/2018:**

I had encounter an issue with implementing a rotation on both the X and Y axis on the same script via; the mouse. This was because “rotation” was already defined to alter the x-axis. To solve this issue, I applied another script onto the camera (instead of the player) specifically allowing y-axis rotation as well, meaning the player could fully rotate the viewpoint using the mouse alone.

Much like the issue with the X/Y-axis rotation, I was having issues defining both horizontal and vertical movement as they originally used the same float, “speed”. It wouldn’t allow me to do this as “speed” was already being used for Vertical movement. To get around this, I created a separate public float named “sidespeed” for horizontal movement, allowing the player to move forward/backwards, left/right **and** enabled diagonal movement.

**Task/Tutorial Journal 3**

**25/10/2018:**

Implementing a line render onto the grappling hook proved troublesome and used up most of the development time for the grappling hook. This is for the simple reason that I am well acquainted with using Raycasts at this time, but I have no experience with using a line render. With this said, I was required to self-teach not only how to implement a line render through the unity editor (and the settings that are able to be adjusted), but also how to reference the line render via; code.

Although the grappling hook is comprised of terms and functions that I am now well acquainted with, coding the grappling hook still proved to be very challenging. The hardest part about coding the grappling hook was refining the momentum gain/decay to create fluent movement.

**Task/Tutorial Journal 4**

**03/11/2018:**

This was one of the smaller tutorials compared to the rest, so the issues found within this tutorial were rather limited as there was not much of a margin for error. The main “issue” I had come across when reconstructing this tutorial was forgetting to assign the “RespawnPoint” to an Empty GameObject, meaning the player wasn’t able to respawn when colliding with a purple cube. Not only this, but I had also forgotten to add an additional box collider to the red cube to act as an external hitbox. This was a problem as the player would often need to forcibly shove themselves into the red cube in order to trigger its effects.