Programming Journal

**18/02/2020**

We begin again, ready to take what we learned and accomplished in semester 1 through semester 2.

We started off by looking at an example ‘formation movement’ package created by our lecturer. I’ll be honest, I have not a clue what was going on.

**03/03/2020**

I have never programmed any kind of low/no gravity simulating things in Unity, and after seeing a piece of work completed by a course-mate I was inspired to do just that. After a little research, I concluded that building an amateur ‘flight simulation’ was my best step.  
It only took me half hour to get it all working, and then a little longer after that to tweak a few things, for example the ‘plane’ clipping through the terrain.

**10/03/2020**

In practice for my AGP I decided to put together my first ‘first person controller’. I feel relatively confident in putting together a third-person controller script, I had no idea that first person would be so different and so much more of a challenge. I so far have my ‘player object’ which consists of an empty, the camera and a capsule collider imitating a body. I am perplexed.

I did a little more research on unity documentation and I now have a player object that can look around with the mouse. My next stage will be clamping the look controls to work in the same way as a human head/neck – meaning I need to stop the look up at a point, and the look down.

**25/03/2020**

I skipped clamping, pushing it down my ‘to-do’ list and headed straight on into movement controls for my first-person controller. I learned about SimpleMove and [SerializeField] and the fact that you must spell it the American way always catches me out. It turned out not to be as difficult as I initially thought so I continued adding a jumping mechanic to the controller, also.

**02/04/2020**

I started work on my ‘Spawner’ package. I find it satisfying to look at when objects just manifest endlessly from a single point, and they can also be used in games that revolve around battling waves, or they can be used as effects. I decided this would be a good one to make.  
I’d written a similar script a little while ago, so I didn’t anticipate any bumps. Of course, I was wrong not to. I was aiming for a gentle spawner throwing cubes up gently into the air with them bouncing down after a couple of seconds. I got a violent surge of cubes going up and never returning down.  
 I knew my script was correct, so I focused on my variable settings and eventually regained control of my erratic spawning. Amending the numbers in my inspector didn’t seem to be having any kind of effect. I *thought* my script was correct, but I had hard coded the upForce value and so this was the cause of my spawner not being what I intended. Easy fix, luckily.

**10/04/2020**

I had some struggle with my first-person controller, so I had left it for a while. I return now having successfully clamped its look controls, though not without some obstacles. For some reason when I first implemented the clamping, it worked with the vertical controls but somehow removed all horizontal looking ability. There were no errors in the script, so I had to go on was the small amount of code I’d just added. After several google searches and checks with friends, I was still empty handed as to what the problem was.  
I shut down unity.  
I reopened unity.  
*The script was now working fine.*

**11/04/2020**

I placed great importance on my first-person controller being perfectly smooth. And so far, it was. Until the introduction of slopes. Going up a slope wasn’t so much the problem but going down caused unattractive jittering that I was determined to fix.  
 I learned about the player travelling faster than unity’s gravity system was able to calculate and so I investigated the use of ray casting to fix this. I was able to set the ray cast origin to the centre of the player and soft coded the measurement variables so that they could be easily adjusted within the inspector till the player could glide down slopes.

**16/04/2020**

I have worked on this inventory system for what feels like my entire adult life. After a substantial amount of professional aid, I have the ugliest, bare bones of an inventory going on, but I remain *absolutely* clueless. I have secured knowledge regarding creating a UI setup for one but in terms of the actual programming…  
My initial attempt consisted of three scripts, where one sat in the middle passing information and references from one to the other – item < inventory > slot. I managed to get it working, however, despite attaching an icon as I was supposed to, no icon showed up within the UI when prompted.  
 So I scrapped it all and started again.  
It wasn’t any easier or any less confusing but it was successful and functioning. It’s something I need to work further on, however, as I don’t feel that I have learned anything despite it working. It remains overwhelmingly confusing to me.

**07/05/2020**

To complete my packages task, I began coming up with a way to put them together into a game. I realised that when doing my individual packages, it would’ve helped to think about them eventually working together, but perhaps it was better that I didn’t and now had to really think outside the box to use them.  
 I decided on a first person game where you had to make it to the end of a hallway without being killed by falling cubes. Three hits and you’re out. Real unique, I know.  
I had to add a couple of functions to get this set up such as player health, and damage but there was a fatal flaw in my set up. To damage the player, the cube needed to find the player. I didn’t realise that a prefab would not be able to reference other scripts or objects, in order to retain its current information and maintain its ability to be used at any point during the game.  
I am yet to fix this problem.