**Day One (19/01)**

* Read over brief. Start from the top, that means doing inputs first.
* **Inputs**, meant to be based on a touch screen. Raycasting a certain area on a canvas and finding the input’s distance from the centre of the ‘ray area’ seems like the best solution. –
* Should I make the player have to always start from the centre before moving out (Joystick resetting each time the player lifts their finger/input stops)? No, unnecessary complication.
* Add Inputs.
* Should they work with touches? Didn’t say it needed to but better be safe than sorry. Add Input based on touches. Use Platform Dependant Compilation to have code distinguish when to use which sections.
* **Problem**: movement stops if player moves finger away from input area (which is quite small). **Solution**: Keep track of the mouse (or finger) if it moves away from the input area. Create a large input area.
* **Problem**: crossing over into the other stick’s input area. **Solution**: Check with raycasting if the input has already been assigned to one of the sticks. If it is, ignore it.
* **Problem**: Inputs detected when starting outside of input area, could cause problems if attempting to touch buttons. **Solution**: Make sure that the first frame of input is from the stick-sprite’s restricted area.
* Connect the input values being received into the player movement script. Use an Instance, easier and makes the two scripts a bit more flexible.
* **Problem**: camera that was parented to player now spins when changing rotation. **Solution**: create a script that sets the position of one object to the position of another during update. Add the option to do so manually as well, disabling update and relying on calls.
* **Creating Weapon**: use inheritance to create a system to incorporate multiple weapons.
* Create Weapon Base class with ‘projectile interval’ and ‘reload’ variables. **Problem**: Possibility of melee weapons. **Solution**: Create ‘Ammo Weapon’ class that inherits from ‘WeaponBase’ class. ‘Reload’ variable to go to ammo weapon class while ‘projectile interval’ variable can be reused as the time between swings while still acting as a projectile interval for Ammo Weapon classes.
* **Problem:** Having too many weapons may cause a slow down in the project due to *if­*-statements checking if the weapon is being used. **Solution:** Have the player call the update of the weapon that they’re using.
* **Projectiles.** Spawning and destroying constantly can be taxing in large numbers. Create an object pool and only create more projectiles when needed and disable the projectiles at the end of their life to be reused.
* **Problem**: Different weapons may need their projectiles moving at different speeds. Eg. Rocket launchers with slow(er) moving projectiles. **Solution**: Create speed variable for individual weapons, remove gravity from projectiles.

**Day Two (20/01)**

* New weapons added (Rocket Launcher and Machine Gun). **Problem**: Reload consuming bullets on weapon switch even if we weren’t reloading during the switch. **Solution**: Check to see if we were reloading during our last usage of the weapon.
* New type of projectile, Rocket projectile. Could modify existing object pool script to hold multiple objects but deemed it too costly when simply duplicating the existing object pool object works just as well.
* Power-Ups added. Power-Ups unlock different weapons and refill ammo for specified weapon.
* Spawner added. Allows Power-Ups to be placed randomly inside of area. **Problem**: ‘Random’ class only has to place randomly for spheres, a box collider is being used. Could create a function inside of ‘Spawner’ class to find a random point inside a box collider but that functionality may be needed elsewhere. **Solution**: Extension Methods, create an ‘ExtensionMethod’ class to house extension functions.
* Having something like a spawner locked for only powerups is a waste, alter script to be modular enough to accept things besides powerups, like enemies or environmental hazards, this makes the ‘Spawner’ class dealing more with placement rather than spawning but that’s okay.

**Day Four (22/1)**

* Used Unity Remote for the first time. Looked up an info-page on Unity website.
* Tested touch controls with Unity Remote. **Problem:** Logic Error. Control scheme for touch controls on mobile functioning strangely. The touch’s end not being detected correctly. **Solution:** Run through code mentally, isolating where the project could be breaking. Find bug and fix code.

**Day Five (23/1)**

* Originally setting the camera’s position whenever the player was moving. This would have been fine but the player slides a bit after they stop receiving input, causing the camera to become off centre and to snap back when the player started moving again. **Solution:** Move call from whenever the player is moving to just in the player’s update loop, the performance drop is negligible and acceptable in this case.