Development diary for final year project

Week 1

For this week i just focused on getting a Github repository set up and ready for development to begin.

Week 2

I began the week by making a start scene with a plane and a capsule to use as the ground and unit respectively. After I added them into the world I added navigation to the plane and a nav mesh agent onto the capsule to be used in a script to help the unit navigate. I then set up; a base unit class which currently will only hold two boolean values for checks. The two boolean variables test if the unit is a player’s unit and if the unit is selected. After that I set up a script for the camera which adds movement to the camera when the player uses the movement keys. The movement works by applying a translation vector to the camera’s transform vector with y being 0 and x and z varying depending on input with movement keys with the value defaulting to 0 when input is not present. The camera class also holds functionality for when the player left clicks and right clicks the mouse. When the player left clicks the mouse the camera does a raycast and tests to see if the ray hits a unit. After this the camera also tests if the unit belongs to the player. If the unit is found to belong to the player it either becomes selected or deselected depending on its current selection state. If the player right clicks then the camera again does a raycast but records the point of impact if the raycast hits an object. After this the code loops through the objects to find any selected units. Any units that are selected gets given the raycast hit point to be their new destination for the nav mesh agent to navigate to.

Week 3

At the start of this week I added an enum to the units for the state that they are in. Right now there are three states for idle, moving and engaging. I have also added a GameObject variable to hold which unit it is currently engaging if the unit is engaged. In the camera I have added code for the selection so that if just clicking anywhere all the units will be deselected unless the player is holding shift. If the player is holding shift they will also be able to select multiple units. After this I moved on to add a test for when the player right clicks so that when the raycasts hits a point it first tests to see if the nav mesh agent can compute a complete path to the point. If the agent manages to complete a path the unit will then know that the point is reachable and will start navigating to that point. The unit will also then change its state to the moving state. There is now also a check inside the unit update testing the distance to the nest destination. This check is so that the unit will know when it reaches its destination and will change its state back to idle.

Week 4

This week I mainly spent time on another assignment so I wasn’t able to do as much on this project. I worked on adding more of a base for the different states. I added a state for engaging an enemy and a state for assisting an ally. I have also added a bullet into the project so that the units can fire upon enemy units and vice versa. I used unity’s invoke function to time a cool down function between shots. This way the units don’t fire too much and the cool down speed depends on a value that can be set within the editor. I have also adjusted the camera to an angle that is easier to see more of the level. I had a bug where the units stopped moving after a bit, at first I thought the unit wasn’t being selected anymore. After some investigation I found out it was that I wasn’t setting the isStopped variable back to false as I thought unity would do that automatically.

Week 5

I have started the week be starting production of the points of interest system, the thought process being that units would be taking points into account and then making decisions on their surroundings. Right now I have a point generating on each unit and a debug sphere drawing on every point of interest. Later on in the week I continued development on the generation of points of interest. I have managed to make groups of units as points of interest. I have achieved this by looping through the units when initially making the points. When one unit is found for a point it loops through units again to see if they are within that first unit’s awareness range. If they are within the range they are added to the list of objects within the point of interest. After developing the grouping I moved on to develop functions for working out the position and radius of the point of interest. The position is set to the middle of the group. This is done by adding all the positions together and dividing by the number of objects in the group. The radius is calculated by finding the largest length between objects in the point and picking the longest length to be the radius. However if there is just one unit in the point the radius will be the awareness range of that unit. I have yet to add functionality for if there is just one object but isn’t a unit. I have also added colour to debug gizmo’s to display green for allies and healing buildings, red for enemies and blue for cover. I have also now added cover and the respective functionality for generating points of interest for them, once again being able to group them up so units would get in between cover objects.

Week 6

I started this week off by looking into fixing this nasty bug where after a couple seconds the units would no longer be able to be selected. It took me a while but after much looking online I found that the issue came from having both a nav mesh agent and a rigid body on the same object. I wanted the physics to still look sensible so I kept the rigid body on but changed the collision mode from discrete to continuous. This seems to have fixed the issue for now. After this I moved on to developing the unit behaviours. I started by making a function that calculates the weighting for point of interest so that the units can decide what action would be the most appropriate. After this I developed the code to test all the weighting of the points which changes a bit depending on what type of point this is. I also added an extra check that changes the weighting based on how far they are from the set position of the unit. After some development i now have units engaging enemies when they have an advantage and backing each other up when outgunned. I also have units going to cover when outgunned to get an advantage. At the end of the week I spent some time refactoring code. I made functions for code that I repeated multiple times so that the code is more organised and understandable.

Week 7

I started the week by developing functionality so that the units can run away if drastically outmatched. For this I utilised the ally score and enemy score which shows how powerful the presence of enemies and allies are. If the enemy score is more than double the ally score they will run away from the enemies. I had a major issue where if a unit was too far from their set position they would not return. I tried debugging to see if the set position was changed at all and kept re running tests to see where the unit would stop going back. After much looking though my code I discovered that the code for returning to position still relied on at least one point of interest being present. I fixed the issue by also placing the return code into the check for if the highest weighted point of interest is null. This way they would still be able to run the function to get back to their original position regardless of isolation.