AI pathfinding

Task 3)

a) The main differences of the grid graph, point graph and navmesh, is the way that they generate nodes. The grid graph generates nodes in a grid pattern, that is it creates square (width *height), this makes it able to update the grid graph during runtime, which gives flexibility and fast updates due to their regular structure. Then navmesh graph creates the data for pathfinding in triangular meshes, this makes it faster than the use of a grid graph since less nodes are created, thus less searching, and also gives out very precise movement. The point graph works differently, since it creates a root (transform object), and the graph will be searched for the children of the root, this is what creates the interconnectivity of the nodes, which allows for a lot of customization and flexibility.
b) When researching and using the unity profiler it was very clear that the main bottleneck for pathfinding is the hardware that your pc is using. In my case which I am using quite an old laptop therefore evidently using older hardware. From the profiler certain methods such as the scanning of the graph took a very long time to process, due to the power of your CPU. There are certain ways that you can optimise your code to reduce these bottlenecks, but they will still always be their.

Task 5)

A search tree is a binary tree where every node within tree has two children, the right sub tree and the left sub tree. Whenever a search is being done it start from the root nodes, and if the data is less than what it is searching for, search the left sub tree, else search the right one. The search tree that is being use by pathfinding algorithms is called a graph search algorithm, where it is searching for a path between two nodes in the graph whilst trying to find the shortest path to the target, this keeps traversing until the target node has been reached. Combat AI normally use the monte carlo search tree, which works by first selecting, which means selecting a node which has the highest possibility to win, than expanding, meaning increasing thee options by creating many children nodes, then simulating, where the move will be performed, and reward the best performing move to go down the tree, finally updating, which mean going back through all the nodes and updating the scores one-by-one through every node. Depending on this information, I do not think that the search tree being used for pathfinding can be used for combat AI, as in pathfinding the algorithm traverses through two children only, whilst in combat ai, it is much more complex. Pathfinding also uses informed search trees, where the ai has prior information about the space before it is searching it, and also uninformed search trees where the ai is trying to find the path with only information about the start node and the nodes adjacent to it.

Task 6)

A* pathfinding VS SimplePath

- Simple path can only generate grid graphs whilst A* supports grid graphs, point graphs, and navmesh graphs. This gives a* the upper hand within this aspect as one has many different options on how he wants to go about building his graph.
- Both of these pathfinding plugins support modification of the graphs in use during run time of the program being used.

- SimplePath is single threaded unlike A* which is multi-threaded, this is a process where multi-tasking can be achieved which saves time since multiple tasks are executed at run time.
- SimplePath uses a multi component approach, which mean every agent requires a lot f different components to work, calls and events, to inform every component what is happening, which can be a bit overwhelming. A* pathfinding is much more simpler to use in this sense and is pretty much a ready to go plugin and is best for a beginner.
- A* pathfinding use tags (layers) to show which are is walkable for your ai and what is not, in the simplepath pathfinder everything is done by code and is not easily accessible as it is with the A* since one can do this from the inspector.

Github Link:

https://github.com/noasulMSD61A/SoftcomputingAssignment1

Reasoning why I did not have 15 commits:

I did not want to fill my github with a bunch of small commits that barely had changes and since the big chunk of the assignment was done within a 2 week time span (as other assignments where due before it), I was doing big parts of coding at the same time, that is why I did not manage 15 commits.