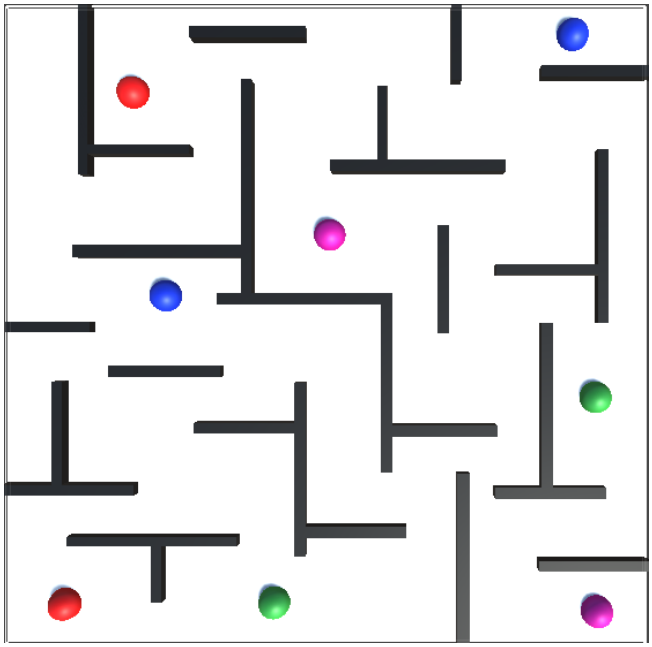
Ismail Ben seddik
Project 1: Pathfinding

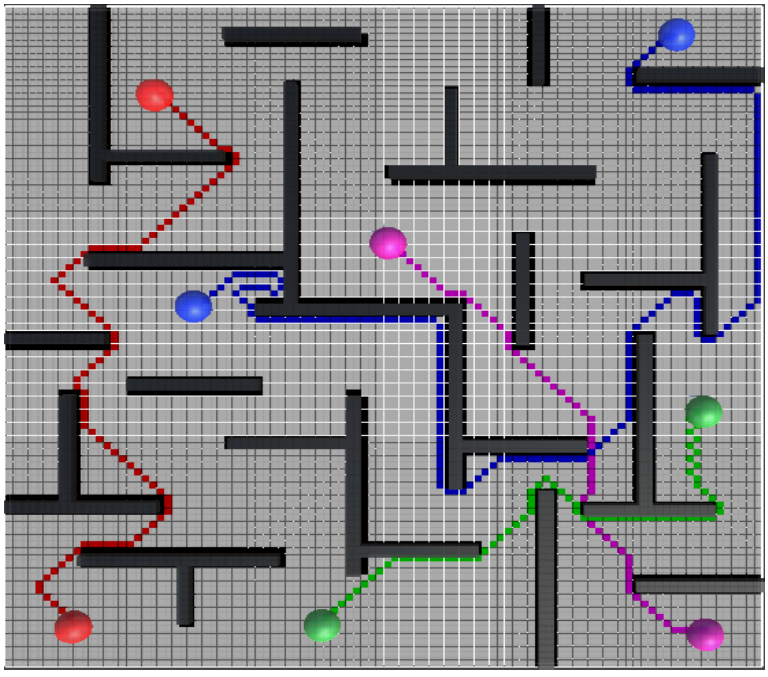
Environment

The following environment was created using Unity:

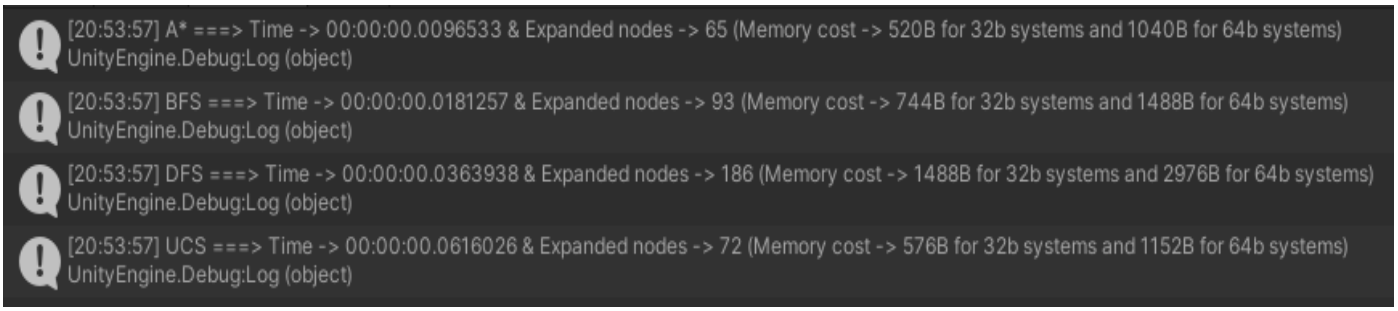


- With only 1 seeker and 1 target for all search strategies' paths get mixed up and the exhibit gets really messy. Hence, strategies' paths, seekers and targets are represented by different colors as: A* (magenta), DFS (blue), BFS (red), UCS (green).
- The environment was deliberately made large and with fairly many obstacles in order to accurately compare the search strategies.

Experiments and Observations

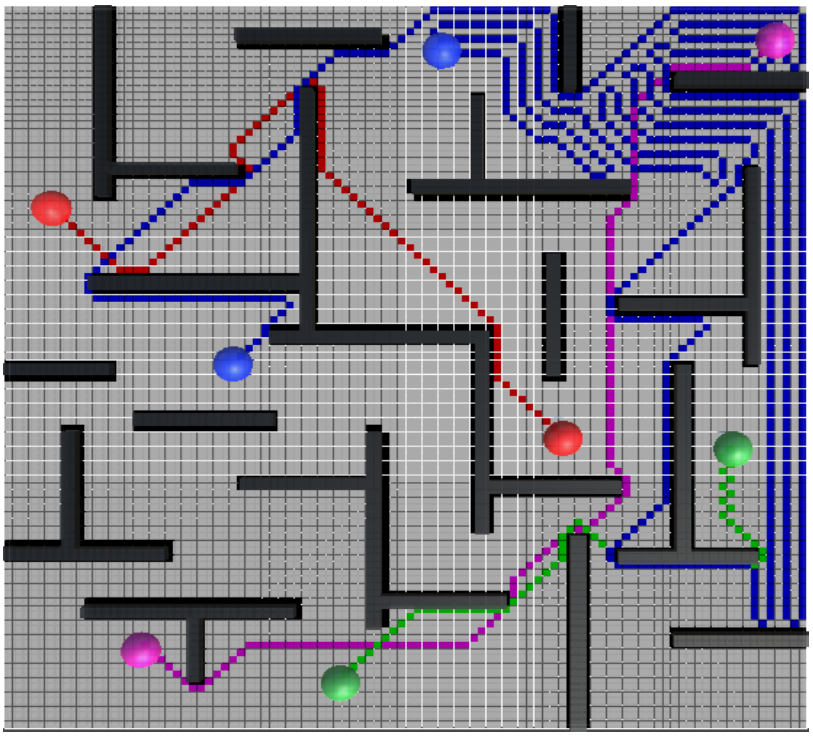


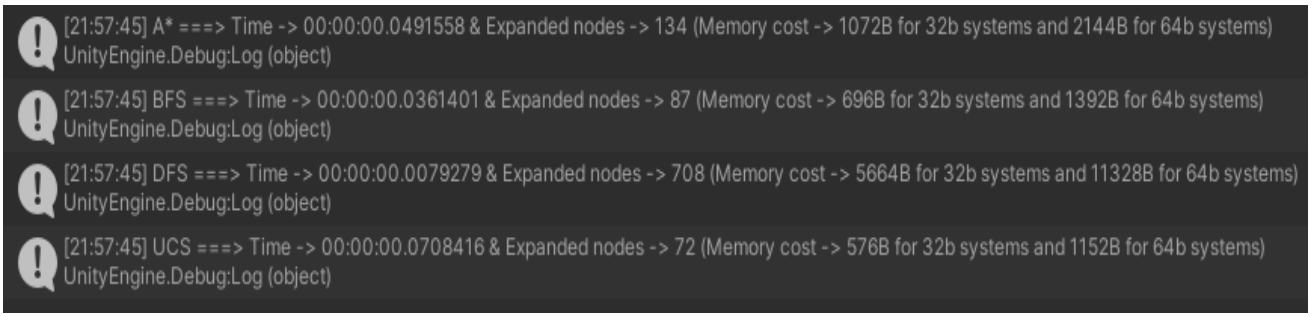
Along with the displayed paths, time, expanded nodes and utilized memory are recorded and logged as follows:



- Through this initial experiment, it is obvious that A* performed so much better than the other strategies in terms of time and expanded nodes (memory). In terms of expanded nodes, UCS follows A* with 72 nodes expanded. Then comes BFS with 93 nodes and DFS with 186 nodes. However, timewise, it is BFS that follows A*. Then, follows DFS and, lastly, UCS. Thereby, on average, we can say that A* is most optimal, then BFS, then UCS and, finally, DFS.

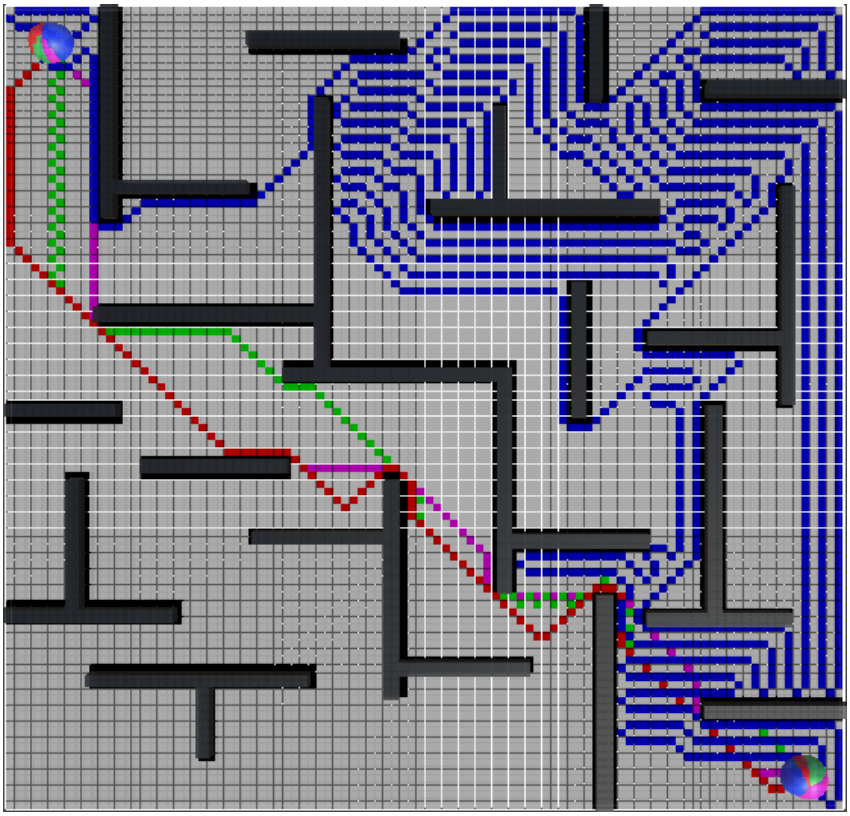
➔ Since A* performed so well, the following is a modified configuration of targets and seekers with an intricate road for A*:

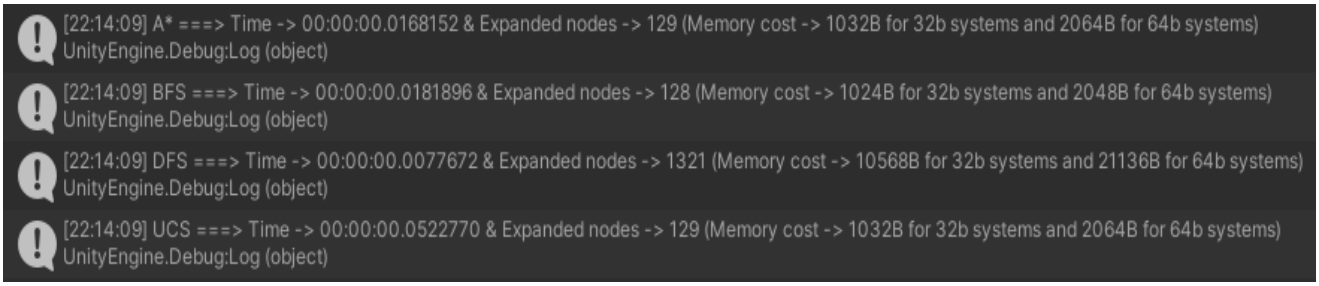




- It is quite obvious that DFS expands the most nodes. With a road full of obstacles for A*, it has done better than UCS timewise although the distance that A* had to traverse is quite superior than that of UCS. DFS, on the other hand, in spite of the short distance between the seeker and the target, has found a very long, clearly non-optimal, path. Still, A* seems to outperform all the other search strategies.

➔ To carry out a fair assessment, seekers and targets of all strategies shall be all positioned somewhat on top of each other, as follows, to give them all the same distance:

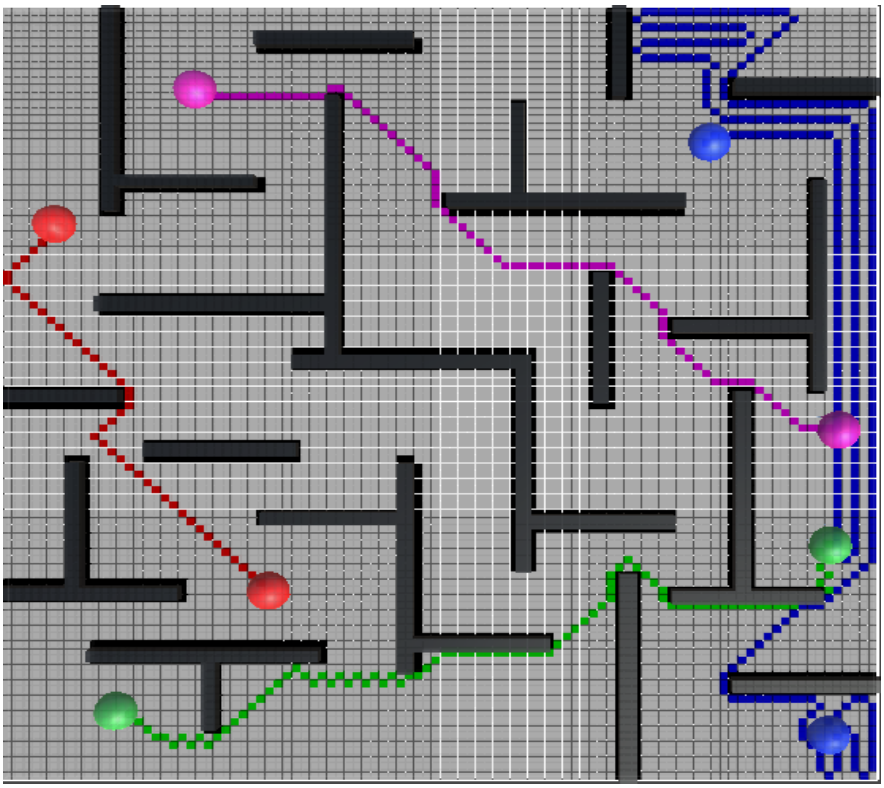




- Again, A* outperformed the other strategies. DFS finds the path in the best time but expands the most nodes (=> uses the most memory). BFS seems to have found the path with good timing and with the least expanded nodes. UCS, however, took the most time to find the path and expanded as much nodes as A*. Although BFS performed really well, it didn't outperform A* and only outperformed UCS and DFS as the target was laid out in breadth rather

than depth. UCS's bad performance is due to computing and comparing costs in the fringe.

➔ The following are the same 3 configurations, but the first one with the normal A* heuristic, the next considers a heuristic that doubles hCost, and the last one computes hCost just like it computes gCost (UCS's cost computation):

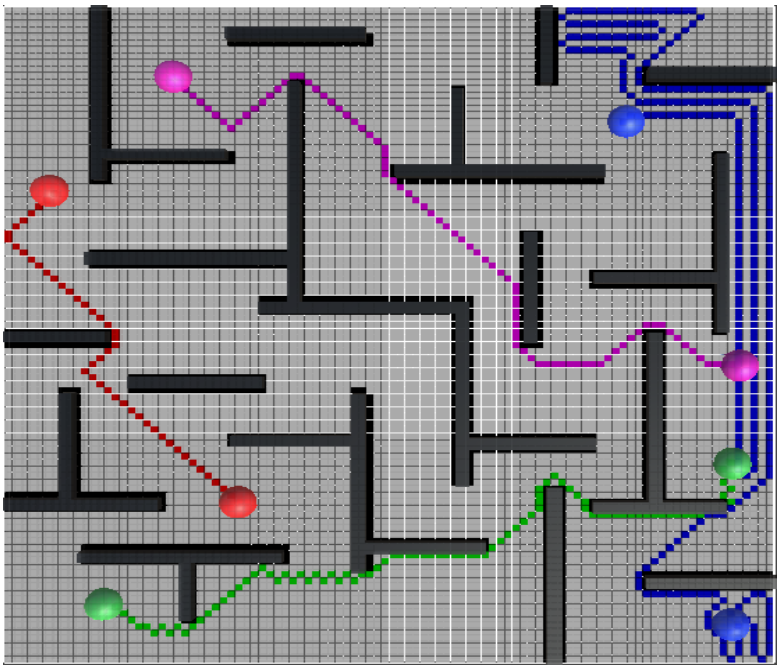


[22:38:41] A* ===> Time -> 00:00:00.0092147 & Expanded nodes -> 80 (Memory cost -> 640B for 32b systems and 1280B for 64b systems)
UnityEngine.Debug:Log (object)

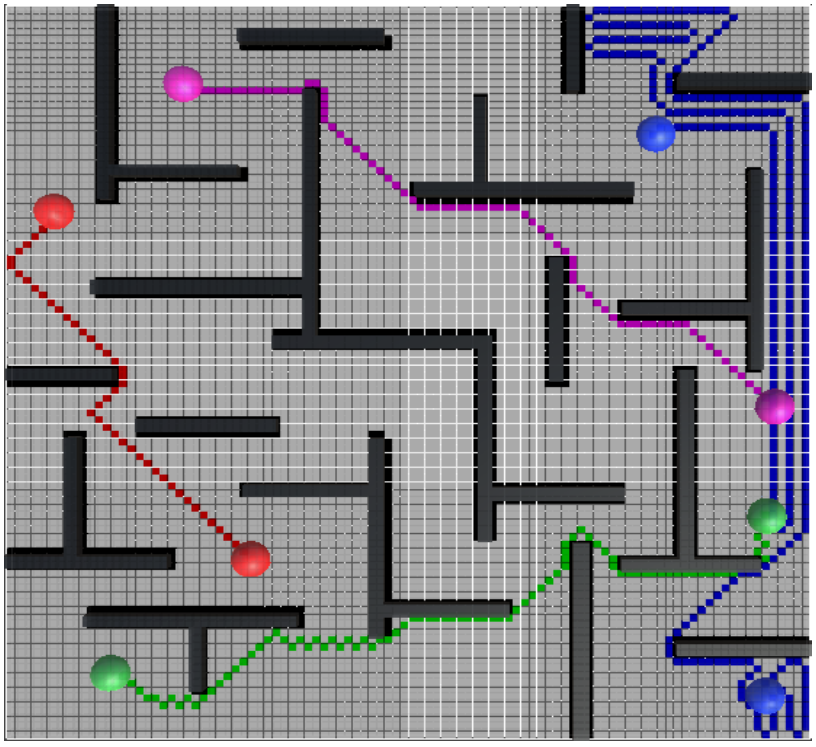
[22:38:41] BFS ===> Time -> 00:00:00.0061214 & Expanded nodes -> 47 (Memory cost -> 376B for 32b systems and 752B for 64b systems)
UnityEngine.Debug:Log (object)

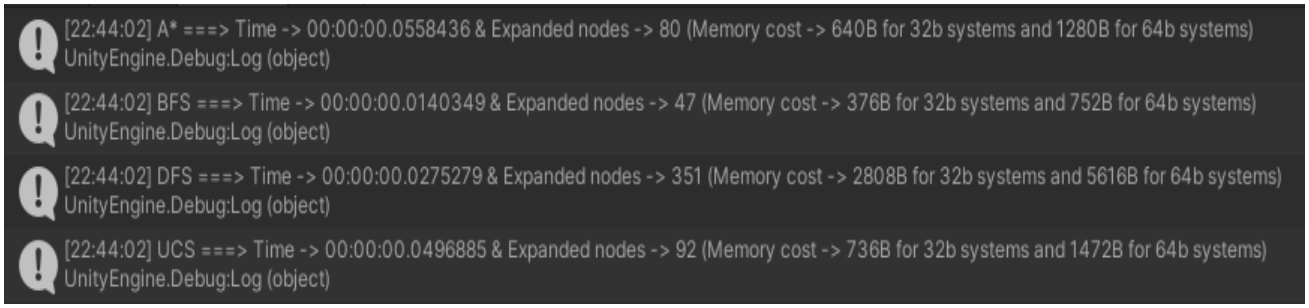
[22:38:41] DFS ===> Time -> 00:00:00.0182931 & Expanded nodes -> 351 (Memory cost -> 2808B for 32b systems and 5616B for 64b systems)
UnityEngine.Debug:Log (object)

[22:38:41] UCS ===> Time -> 00:00:00.0297507 & Expanded nodes -> 92 (Memory cost -> 736B for 32b systems and 1472B for 64b systems)
UnityEngine.Debug:Log (object)



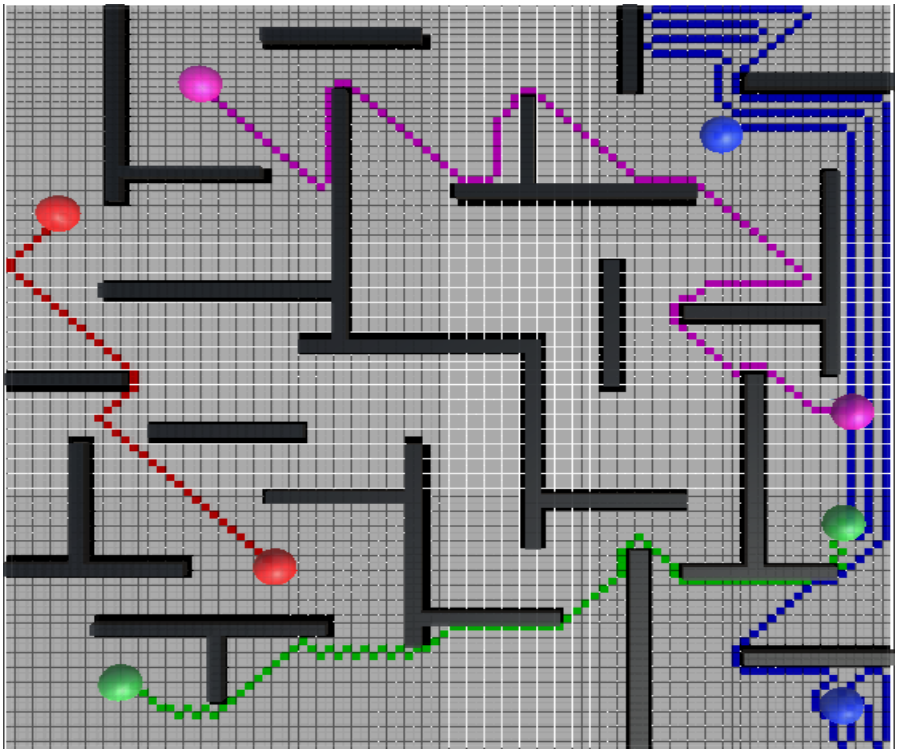
```
[22:41:46] A* ===> Time -> 00:00:00.0070346 & Expanded nodes -> 86 (Memory cost -> 688B for 32b systems and 1376B for 64b systems)
UnityEngine.Debug:Log (object)
[22:41:46] BFS ===> Time -> 00:00:00.0103625 & Expanded nodes -> 47 (Memory cost -> 376B for 32b systems and 752B for 64b systems)
UnityEngine.Debug:Log (object)
[22:41:46] DFS ===> Time -> 00:00:00.0285180 & Expanded nodes -> 351 (Memory cost -> 2808B for 32b systems and 5616B for 64b systems)
UnityEngine.Debug:Log (object)
[22:41:46] UCS ===> Time -> 00:00:00.0425642 & Expanded nodes -> 92 (Memory cost -> 736B for 32b systems and 1472B for 64b systems)
UnityEngine.Debug:Log (object)
```

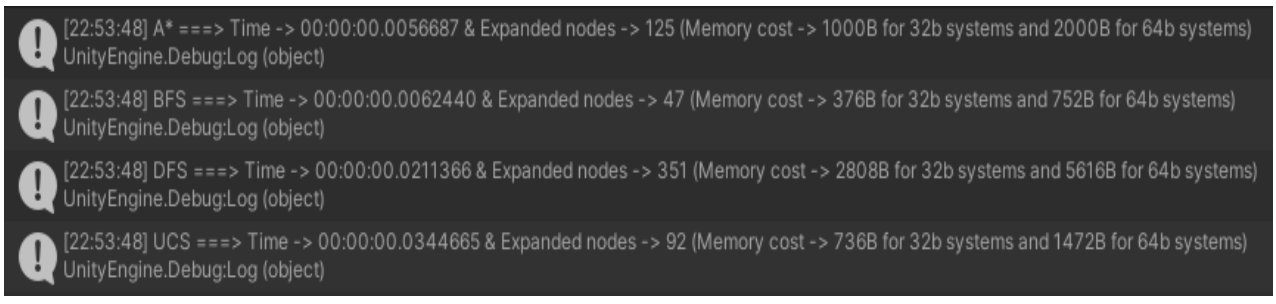




- The results shown above demonstrate a decrease in performance for A*; in terms of expanded nodes (first heuristic) then timewise (second heuristic). Timewise, for the second modified heuristic, A* was the slowest despite the relatively short distance between the seeker and the target.

➔ The following shows the same configuration of A* with a heuristic equal to 0:





- Evidently, A* keeps declining in performance with a heuristic equal to 0. This only validates that A* is optimal only with an admissible heuristic that is closest to the actual least cost to the target.

➔ Yet, with an admissible heuristic, A* remains the most optimal performer among the other strategies. UCS can be seen as the least optimal, on average, in terms of time as well as in terms of memory consumption. However, even though BFS and DFS may seem to outperform UCS, but they don't actually compute the cost (distance from the start state). So, we can say that both A* and UCS are optimal as they both consider the costs leading to the target. However, A* is more optimal than UCS as it not only considers the distance from the start state (gCost) but also the distance from the current state to the goal state (hCost), which yields a better outcome for A*. Generally, DFS seems to outperform BFS timewise, whereas, BFS outperforms DFS memory-wise, as DFS expands more nodes (going down the state search tree and exploring it in depth). That said, all strategies are complete as each seeker finds its corresponding target regardless of its position.

References

- Sebastian Lague's Youtube videos: https://www.youtube.com/watch?v=-L-WgKMFuhE&list=PLFt_AvWsXl0cq5Umv3pMC9SPnKjfp9eGW
- Sebastian Lague's code: https://github.com/SebLague/Pathfinding