Une image contenant texte

Description générée automatiquement

Artificial Intelligence

Pathfinding Project

Oumayma El Ghamrasni

Supervised by Dr. Tajjeeddine Rachidi

February 13th, 2022

The analysis will be done based on Unity’s simulations to see different pathfinding algorithms running simultaneously along with the amount of time taken to run them and the number of nodes explored. In each experiment, I ran only 4 pathfinding algorithms and then added DFS in the next experiment using the same environment. I did that only because it covers a large space which makes it difficult to make observations. First, I would like to mention what color correspond to each path: Cyan: DFS, Yellow: BFS, White: UCS, Magenta: A\* Euclidien, Green: A\*Manhattan. For the seeker and the target, the yellow node is for the seeker and the red one is for the target.

Environment 1:

Une image contenant texte, tableau de points

Description générée automatiquement

*Figure 1: BFS, UCS, A\* Euclidien, and A\* Manhattan pathfiding simulation*

In the first test, I made the experiment using only 4 paths (BFS, UCS, A\* Euclidien, and A\* Manhattan) to be able to see most of the path since DFS covers a large space and hides the other paths. In figure 2, I will show the whole pathfinding algorthims along with their analysis. In this case, we can see that BFS is hidden (The yellow color is not present).

Une image contenant texte, équipement électronique, afficher, capture d’écran

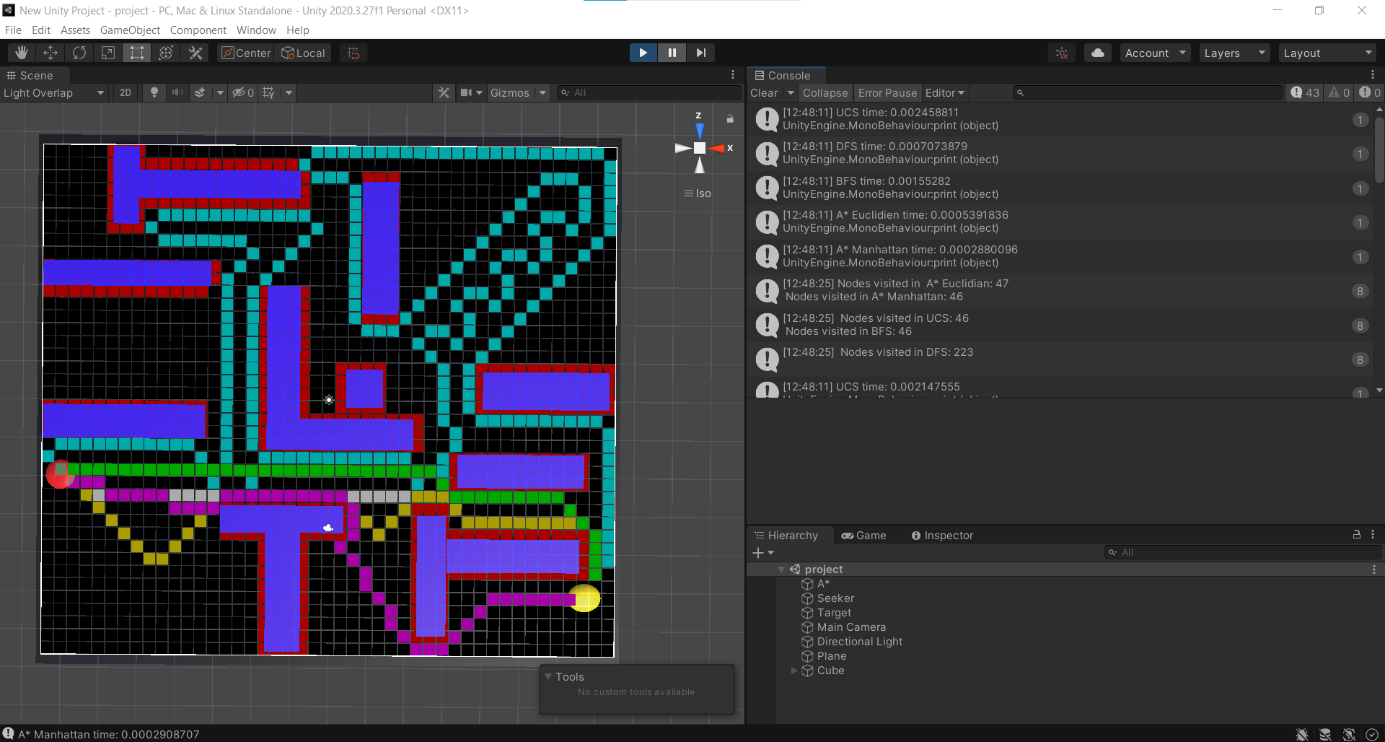
Description générée automatiquement

*Figure 2: BFS, UCS, DFS, A\* Euclidien, and A\* Manhattan pathfiding simulation*

Here we have the same scenario as the previous one but I included all the paths.

For DFS, as I expected, it has the greatest number of nodes explored since it goes in depth. It explored 440 nodes compared to BFS, UCS, and A\* Euclidien that explored only 43 and A\*Manhattan that explored 46. I expected DFS to consume more time since it needs to explore a big number of nodes. However, the simulation shows that it takes approximately 3ms. BFS takes about 1ms, UCS takes about 2ms, A\* Euclidien takes 0.6ms and A\* Manhattan takes 0.5ms. We can conclude from this analysis that A\* Manhatten and A\* Euclidien outperformed the other algorithms. So, in this case, the A\* implementations were an advantage based upon the time they took to finish. Even though the difference is not big between both A\* Euclidian, A\* Manhattan, and BFS, I thought that BFS will be faster since it requires less computations. Also, for memory space, I thought that BFS will explore more nodes since it explores the shallowest nodes. However, regardless of how optimal UCS is, it took more space (47 explored nodes vs 43). Therefore, we can say that BFS is optimal as well in this case since the cost is 1.

Environment 2:



*Figure 3: All pathfiding algorithms simulation with a seeker and a target in a close depth level*

In this second experiment, I tried to put the seeker and the target in a place that will allow me to see all the pathfinding algorithms clearly. As we can see in the simulation, both UCS (White), A\* Manhattan (Green) take a straight path to the goal. They explored the same number of nodes which is 46 in a period of time of 2.5 ms and 0.3 ms respectively.For BFS (Yellow), even thought it did not take a straight direction but it explored the same number of nodes as UCS and A\* Manhattan (46 node) in an amount of time of 1.5 ms. A\* Euclidien (Magenta) was alternating between straight moves and diagonal ones and thus I expected it to explore more nodes than A\* Manhattan, BFS, and UCS. However, it explored only 1 additional node (47 nodes in total) in 0.5 ms. For DFS, it did a deep exploration of nodes. It explored 226 nodes in only 0.7 ms.

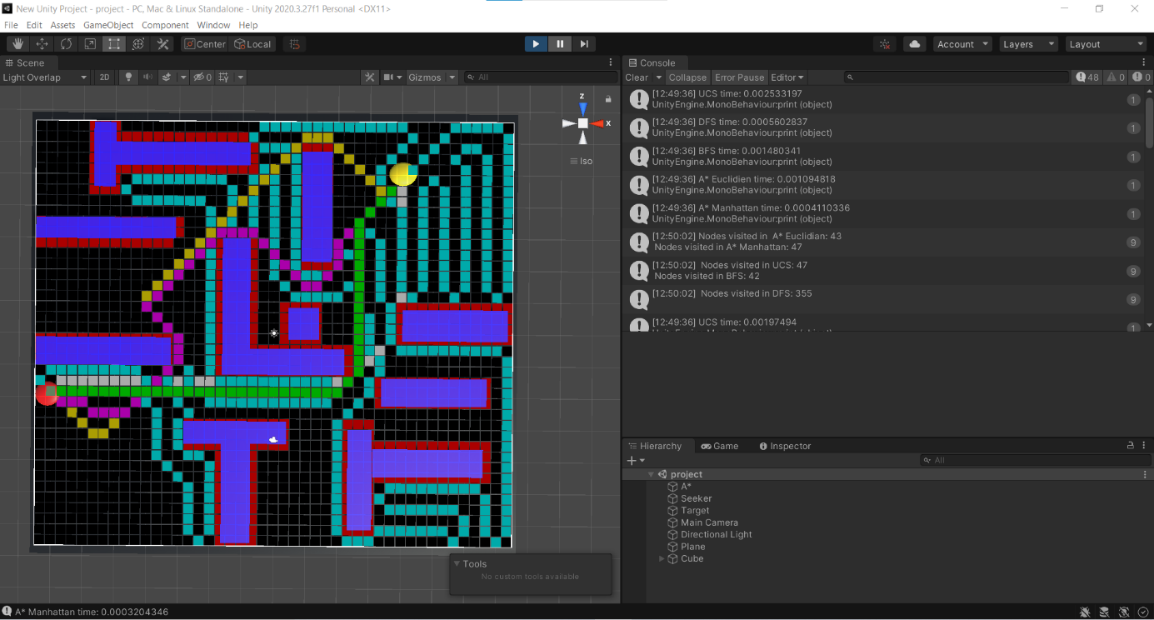
By comparing this data, we can conclude that all the pathfinding algorithms are complete but they are not all optimal. A\* Manhattan is the fastest algorithm to reach the goal (0.3ms). Even though it explored the same number of nodes as UCS, the latter spent the largest amount of time to reach the goal (2.5ms). DFS to end up being the third fastest algorithm in this case. This is due to the fact that we are searching for a target that is deep and far from the seeker. Finally, we can say that A\* with Euclidien and Manhattan heuristics and DFS outperformed BFS and UCS.

Environment 3:

Une image contenant texte, tableau de points

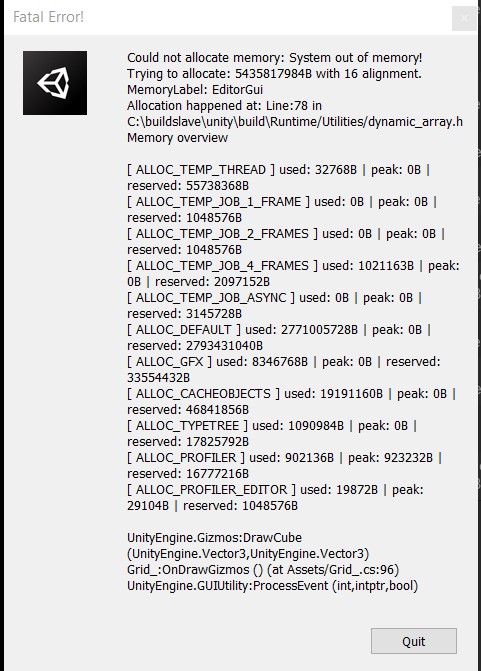
Description générée automatiquement

*Figure 4: BFS, UCS, A\* Euclidien, and A\* Manhattan pathfiding simulation with seeker and target that are far from each other.*



*Figure 5: All pathfiding algorithms simulation with seeker and target that are far from each other.*

In this scenario, I put the target far from the seeker and run all the pathfinding algorithms. All the pathfinding strategies were complete. I expected DFS to win BFS in terms of time needed to reach the goal and that was the case. DFS explored 355 nodes in approximately 0.6 ms while BFS explored 42 nodes in 1.5 ms which makes sense since going deeply is better than exploring the shallowest node in this case. A\* Euclidien explored 43 nodes in 1 ms which makes DFS faster. Also, A\* Manhattan explored 47 nodes like UCS, but UCS took 2ms compared to A\* Manhatten that spent 0.4ms to reach the goal. A\* Manhatten was the fastest among all the other algorithms but it was surprising to see that DFS was the second fastest algorithm. However, this can be due to the fact that UCS needs to do calculations like gcost of all the neighboring nodes to choose the cheapest cost compared to DFS that just explores the deepest nodes.



*Figure 6: The message displayed as a result of running the pathfinding algorithms in a big environment.*

In this case, I tried a bigger environment with a bigger plane and multiple obstacles. However, in such a big environment with 5 pathfinding algorithms the system crashed. I received different error messages including the one shared in figure 6. In this message, we can clearly conclude that the pathfinding algorithms need more space in memory than the one allocated. Thus, the system ran out of memory and then crashed. Another message was related to the GPU where it states that there is not enough space in the memory to allocate for the GPU.

Conclusion:

To conclude, depending on the environment we tackled, the most efficient of the 5 algorithms is A\* Manhattan. Also, A\* Euclidien had a difference of 0.1 ms compared to it in the timinig it took in both environment 1 and 2. Thus, both A\* with Manhattan heuristic and Euclidien heuristic are more optimal than the other algorithms. The one that was the least efficient in two environments was UCS while DFS was the least efficient only in the case where the target was close to the seeker. BFS on the other hand had a short execution time since it explores the shallowest nodes until it reaches the target. Therefore, even though most of the times it explores the same number of nodes as A\*, it spends more time because it looks blindly for the goal. Finally, A\* with its two different heuristics was complete and optimal compared to the other pathfinding strategies.

References

Sebastian Lague Youtube channel : <https://www.youtube.com/watch?v=-L-WgKMFuhE&list=PLFt_AvWsXl0cq5Umv3pMC9SPnKjfp9eGW>

Github course code: <https://github.com/SebLague/Pathfinding>