Assignment 1: Unity shortest path using (UCS, BFS, A\*, DFS)

Mohammed Khalil Ghali: 76368

Link to my GitHub Repository:

https://github.com/khalil-ghali/PathFinding\_Project

* DFS: Blue
* BFS: Yellow
* UCS: Green
* A\* (video heuristic): Cyan
* A\* (Manhattan heuristic): Grey
* A\* (Euclidean heuristic): Magenta

|  |  |
| --- | --- |
|  | **References: { Sebastian Lague Youtube channel and github code** |
|  | **https://www.youtube.com/watch?v=-L-WgKMFuhE&list=PLFt\_AvWsXl0cq5Umv3pMC9SPnKjfp9eGW** |
|  | **https://github.com/SebLague/Pathfinding** |
|  | **}**  **Info about functions and related files:**  FindPathA: finds shortest path using A\*mand the video heuristic FindPathAEuc: finds shortest path using A\*mand the Euclidean heuristic. FindPathAMan: finds shortest path using A\*mand the Manhattan heuristic. FindPathUCS: finds shortest path using uniform cost search. FindPathBFS: finds shortest path using Breadth first search. FindPathDFS: finds shortest path using Depth first search. ----- RetracePathA,RetracePathAEuc,RetracePathAMan: marks the nodes that constitue the shortest path of A\* depending on the coresponding heuristic. RetracePathUCS: marks the nodes that constitue the shortest path of A\*. RetracePathBFS: marks the nodes that constitue the shortest path of A\*. RetracePathDFS: marks the nodes that constitue the shortest path of A\*. ----- GetDistance: Video heuristic gets the optimal distance between the two parameters of the function considering that some directions have a higher cost that normal movement GetDistanceSQRT: Euclidean heuristic. GetDistanceManhattan: Manhatan heuristic. -------- File containing colorisation and grid structure: Grid.cs File containing the node structure: Node.cs File containing the scenes from screenshots: scenes |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |  |
|  |  |

I’ll start the report by a screenshot of different algorithms using different seekers and targets. A screenshot of a computer

Description automatically generated with medium confidence

But this screenshot can not serve to compare different algorithms while the next snapshots shall serve this purpose.

This Snapshot contains A\* with its different heuristics: (same seeker and target)Graphical user interface

Description automatically generated

As we can see her A\* using Euclidean heuristics did better in terms of runtime while they did the same in terms of memory usage(nodes visited)

A\* Manhattan is better in terms of the time efficiency but it rather explores more nodes

This Snapshot contains:

* A\* using the videos heuristics
* A\* using Manhattan heuristics
* A\* using Euclidean heuristics
* UCS
* BFS

A screenshot of a computer

Description automatically generated with medium confidence

We can see that A\* Euclidean and Manhattan heuristics were the best in terms of run time while A\* with Sebastian’s video heuristics and UCS were almost the same

This Snapshot contains DFS in addition to all the previous algorithms:

A screenshot of a computer

Description automatically generated

In this screenshot we explore the case where DFS is almost the most optimal algorithm in terms of time and not the memory usage while in the next screenshot we will explore the case of DFS terrible path due to exploring deep nodes:

A screenshot of a computer

Description automatically generated with medium confidence

-UCS can also have a better run time than DFS in some situations(we used the same start and finish to compare)

A screenshot of a computer

Description automatically generated with medium confidence

General Remarks:

-When we change A\* heuristics the path changes depending on the new calculated distance, We tried multiplying the distance to come up with a new heuristics and the A\* of the video did better while when we tried a totally different heuristics as Manhattan and Euclidean A\* we noticed a better run time and memory usage

-DFS explores the deepest nodes until it finds the solution which explains the long blue lines that covered everything, it also uses a lot of memory since it explores all the nodes in a tree until it finds the result, but it did good in terms of runtime

- BFS gives a very good path but not the best in term of optimality

-UCS time can be same as DFS time and same as BFS space

-UCS can be better if A\* heuristics are not good enough

-A\* is the best when used with good heuristics as we seen with Manhattan and Euclidean heuristics and even sometimes with Sebastian’s heuristics.