

Computer Games Development SE607

Software Functional Specification

Year IV

|  |  |
| --- | --- |
| Donal Howe  C00249662 | [Student Name]  [Student Number] |
| [Date of Submission] | |

[Declaration form to be attached]

Contents

[Acknowledgements 2](#_Toc54713736)

[Functional Specification 2](#_Toc54713737)

[References 2](#_Toc54713738)

# Acknowledgements

We would like to thank the following people who assisted in completing this project including;

John Doe of ACME who kindly agreed to …

I would also like to thank #### Solutions for use of ….

# Introduction

The objective of this project was to put together a comprehensive comparison of guided and non-guided based pathfinding algorithms to the incremental dynamic pathfinding algorithm known as Dstar Lite under a game’s development context. So one can decide based of the information shown in this document whether to or not implement Dstar lite into their project or perhaps to implement another algorithm such as life long planning A star or, A star itself.

Description of D star Lite

Dstar Lite works as a dynamic A star where it can make changes to the path along the graph without having to rerun the process of calculating the path. Where Astar has to calculate the heuristic (cost of the node form the destination + the cost of the node from the beginning node) for each node upon running the algorithm to find the shortest path D star does not. It works by only investigating nodes which have been affected by a non-traversable which has been placed on the path. This in turn makes rerunning the algorithm potentially cheaper than having to recalculate the entire path.

Description of A star

Astar is a heuristic algorithm being that it knows the end and start point. It then tries to find the shortest path to the end point, however it will rerun itself if an obstacle gets in the way. Astar can find the shortest path through a priority queue which will compare the values of each node using both their Hcost( distance from the node) and Gcost( distance from the start node). This is how it knows to look at certain nodes first.

Description of Dijkstras algorithm

Dijkstras search algorithm is a guided search algorithm that uses node weights and connections to find the shortest path to the goal node. Whereas Astar uses the heuristic value distance from the goal node as hcost and distance from the start node Gcost to find the path, Dijkstras only uses the distance from the start node of each node to find compute the shortest path

Description of Lifelong planning Astar

Lifelong planning Astar is an incremental pathfinding algorithm that finds the path by updating the gcost of nodes from previous searches rather than recalulating the entire graph it is one step down from Dstar Lite which is a continuation of the lifelong planning astar algorithm.

Description of Depth First Search

Depth first search is an example of a non- heuristic guided search algorithm, it starts at the root of the graph in the case of this project being the start node which you select. It then traverses through the graph using the neighbours so eventually find the goal. It is not guided it simply goes as far as it can given a specific direction chosen.

Description of Jump Point Search

Functional Specification

The software will in essence function as a visual pathfinding application. So the user will run the application and see a basic grid they can then adjust the size of the grid to three specified sizes “Small” being a 10x10 grid, “Medium” being a 50x50 grid and “Large” being a 100x100 size grid. They can then choose from a variety of pathfinding algorithms them being Astar, Dstar Lite, Dijkstra’s algorithm, lifelong planning Astar , jump point search and the only no heuristic pathfinding algorithm depth first search. The user can also place down obstacles during process of the algorithms search and before the algorithm has been ran if they perhaps are looking for a specific path, onto the grid which will have the pathfinding algorithms react to them and find a corresponding path.

**Algorithm Table and Grid Sizes**

|  |  |
| --- | --- |
|  | The Menu:  The user can select grid size and the algorithm they wish to use. |
| A picture containing shoji, building  Description automatically generated | Grid size “Small”:  The small grid with 100 cells and row of columns of 10 each |
| Background pattern  Description automatically generated | Grid size “Medium”:  The medium grid with 2500 cells and row of columns of 50 each |
| Background pattern  Description automatically generated | Grid size “Large”:  The large grid with 10,000 cells and row of columns of 100 each |

**Console Visualisation: (ScreenShots from the console)**

|  |  |
| --- | --- |
|  | The Astar Search algorithm being ran and displayed in the console there will be a visual representation in the application. |
| Text  Description automatically generated | The Lifelong Planning Astar (LPA\*) algorithm being ran and displayed in the console there will be a visual representation in the application. |
| A picture containing text  Description automatically generated | The Jump Point Search algorithm being ran and displayed in the console there will be a visual representation in the application. |
|  | The Dstar Lite Search algorithm being ran and displayed in the console there will be a visual representation in the application. |
| Text  Description automatically generated | The Dijkstra’s Search algorithm being ran and displayed in the console there will be a visual representation in the application. |
| Text, chat or text message  Description automatically generated | The Depth First Search algorithm being ran and displayed in the console there will be a visual representation in the application. |

**Code visualisation: (code snippets)**

|  |  |
| --- | --- |
| Text  Description automatically generated  Text  Description automatically generated  Text  Description automatically generated  Text  Description automatically generated | The initilise portion of the Lifelong Planning Astar fuction  Compute shortest path function.  Part A  Compute shortest path function.  Part B  The update node function which based on whether it is in the queue and if the node is not equal to the goal node |
| Text  Description automatically generated  Text  Description automatically generated  Text  Description automatically generated | The dijkstras algorithm function which returns the shortest path based of a nodes G cost  Part A  Part B  The g cost comparer for Dijkstra’s which compares the Gcost of two separate nodes |
| Text  Description automatically generated | The depth first search function which returns a path by following whatever node is next |
| Text  Description automatically generated | My heuristic function which calculates the distance from a given node to another. It is called by passing through the end node and the current node |
| Text  Description automatically generated | The calculation of the D star key |
| Text  Description automatically generated  Text  Description automatically generated  Text  Description automatically generated  Text  Description automatically generated | The Main Dstar lite function which controls the wall handling and progression of the robot position on the path  The compute shortest path function which controls the handling of the different types of inconsistent nodes being over consistent and under consistent.  The initialise Dstar function initialises all of the nodes in the grid to suit for the algorithm to work and pushes the goal node into the priority queue with the correct RHS cost and Gcost  The Dstar Lite Update Vertex function which updates the key costs of nodes and potentially pushes them into the queue for further investigation. |
| A screenshot of a computer screen  Description automatically generated with medium confidence | My Astar function compute shortest path function |

**Visualisation of Paths using the different algorithms available.**

|  |  |
| --- | --- |
| A picture containing shoji, building, silhouette  Description automatically generated | an example of basic walls. Red Nodes are the walls placed on the grid. The grid size in question is the small grid of size 100 nodes. |
|  | Path returned using Astar Search algorithm on a grid size “Small”  Green Node = Start Node  Magenta Node = Goal Node |
|  | Path returned using Dstar Lite Search algorithm on a grid size “Small”  Magenta Node = Start Node  Blue Node = Goal Node  Dstar Lite with Debug on second Screen |
|  | Path returned using Lifelong Planning Astar Search algorithm on a grid size “Small”  Green Node = Start Node  Magenta Node = Goal Node |
| Table  Description automatically generated with medium confidence | Path returned using Depth First Search algorithm on a grid size “Small” |
| A picture containing shoji, crossword puzzle, building  Description automatically generated | Path returned using Dijkstra’s Search algorithm on a grid size “Small” |

**Data Collection visualisation**

|  |  |
| --- | --- |
|  | Dstar lite Small grid data |
|  | Dstar lite medium grid data |
|  | Dstar lite large grid data |

|  |  |
| --- | --- |
|  | Astar small grid data |
|  | Astar medium grid data |
|  | Astar large grid data |

|  |  |
| --- | --- |
|  | Dijkstra’s small grid data |
|  | Dijkstra’s medium grid data |
|  | Dijkstra’s large grid data |

|  |  |
| --- | --- |
|  | Lifelong Planning Astar large grid data |
|  | Lifelong Planning Astar medium grid data |
|  | Lifelong Planning Astar large grid data |

**Design and describe how the application will be used:**

The user will be met with a screen 1 of the 3 screens. They are given the option to choose the size of the grid which they want to place the algorithm on and the algorithm itself which they want to use. They can choose from each of the algorithms and change them whenever they want by simply just selecting another algorithm.The user can then select whether or not they want to race the algorithms. If they select the option to do so another screen will appear where their start and end poisitions chosen will be mirrored. They will then see Dstar Lite only race against their chosen algorithm. They can then choose whether or not to turn on the debug option what this will do is allow the user to see the individual costs of cells such as their “RHS” (Right hand side) cost their “Gcost”(distance from the start position) and their key values ( a calculation necessary to Dstar Lite) on each individual cell and will see these values change as the algorithm progresses. They toggle this on and off as they wish. This debug visualisation is only available on the small sized grid as its to obscure to see as the grid size increases.However they can race the algorithms on each grid size.

# References

<https://core.ac.uk/download/pdf/235050716.pdf> - Path Planning Algorithm using D\* Heuristic Method Based on PSO in Dynamic Environment Firas A. Raheema \*, Umniah I. Hameedb

<https://medium.com/@nicholas.w.swift/easy-a-star-pathfinding-7e6689c7f7b2> -Nicholas Swift Feb 27 2017

<http://www.cs.cmu.edu/~ggordon/likhachev-etal.anytime-dstar.pdf> - Maxim Likhachev† , Dave Ferguson† , Geoff Gordon† , Anthony Stentz† , and Sebastian Thrun‡

<https://www.ri.cmu.edu/pub_files/pub3/stentz_anthony__tony__1994_2/stentz_anthony__tony__1994_2.pdf> -Anthony Stentz

Koenig, S. and Likhachev, M. (n.d.). *D\* Lite*. [online] Available at: <http://idm-lab.org/bib/abstracts/papers/aaai02b.pdf>

encyclopedia.pub. (n.d.). *Jump Point Search Algorithm*. [online] Available at: <https://encyclopedia.pub/entry/24246>