

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 is to hopefully explain the benefits to both A star and Dstar and then go on to show the differences between the two and hopefully supply the reader with enough information and test cases for them to decide which algorithm is most suitable for use inside of their, game or application.

Description of Dstar

Dstar is in essence dynamic Astar and what this means is that it can dynamically change the most suitable path of for instance an obstacle falls onto the path. Where Dstar will simply just deal with the obstacle and go around it, Astar will have to rerun itself which in turn makes it potentially more inefficient than Dstar. Then also one has to consider the complexity of both algorithms and memory overhead.

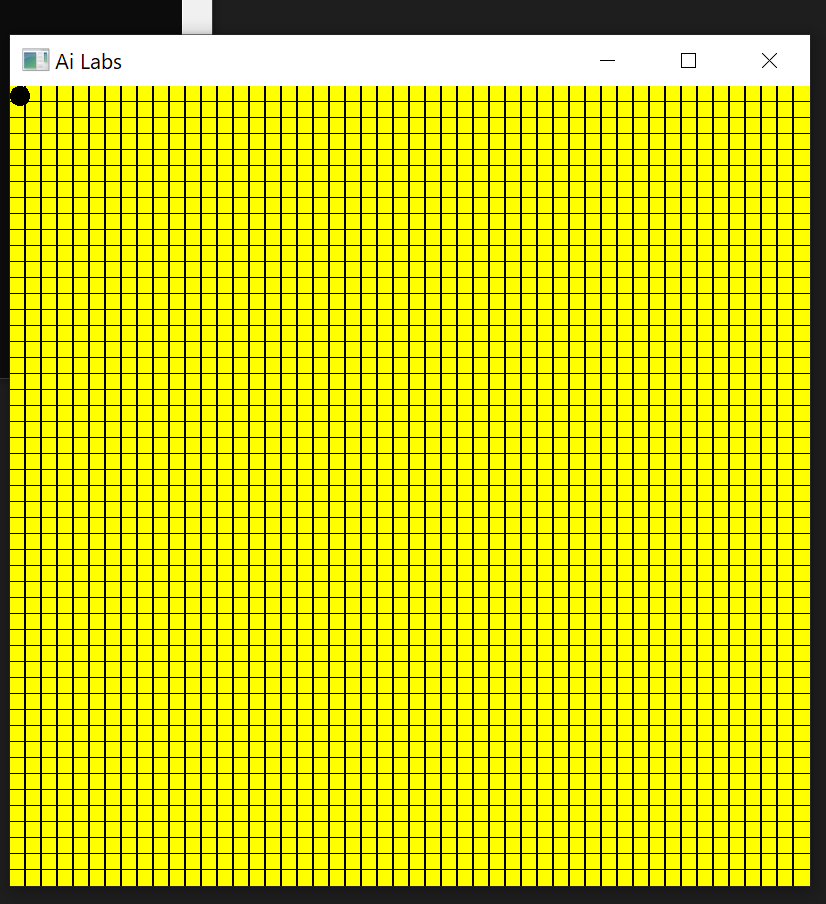
Like Astar , Dstar is a heuristic pathfinding algorithm where it knows its start point and end point and acts just like Dstar up until the point of interaction with an obstacle this is where Dstar comes into its own. This is achievable as Dstar uses two separate functions called raise and lower, where the raise function is used to raise the cost of and arc of a node,( points associated with moving across a node) so it can inturn change the cost of an intraverseable node , and lower does the opposite it will lower the cost of a node for instance if an obstacle dissipates and the user can traverse over the node.

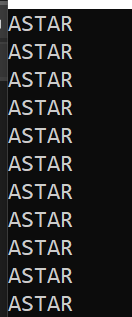
Description of Astar

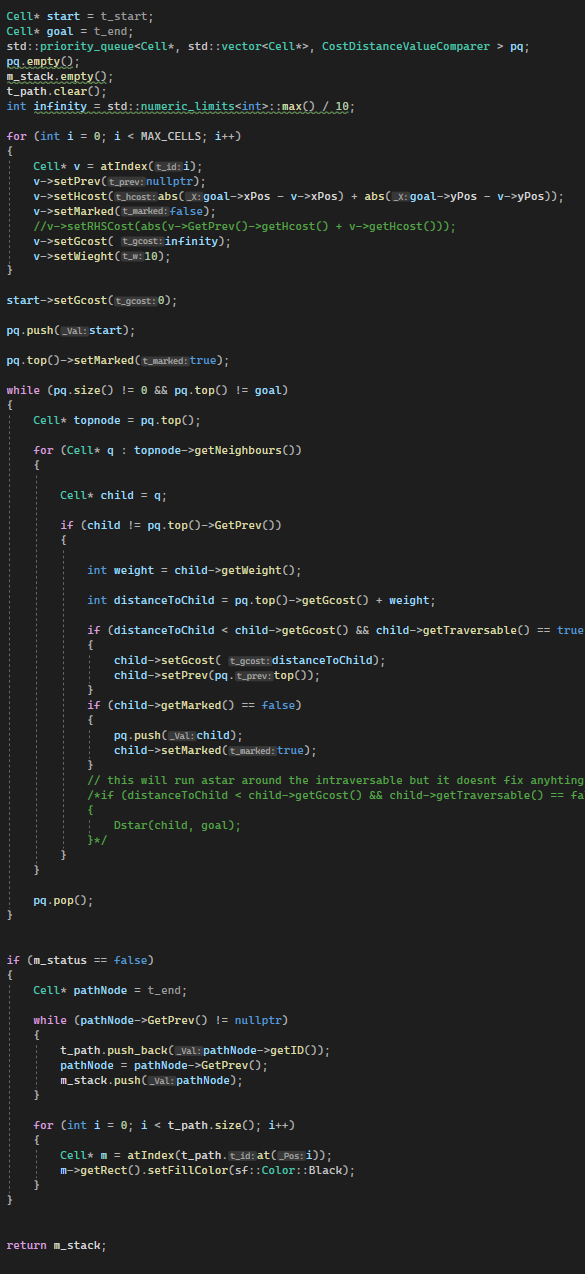
Astar is a heuristic algorithm being that it knows the end and start point. It then tries to find the shortest path to the end point, however it will rerun itself if an obstacle gets in the way. Astar can find the shortest path through a priority queue which will compare the values of each node using 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.

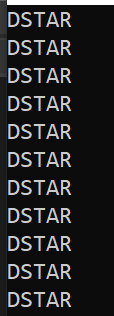
# Functional Specification

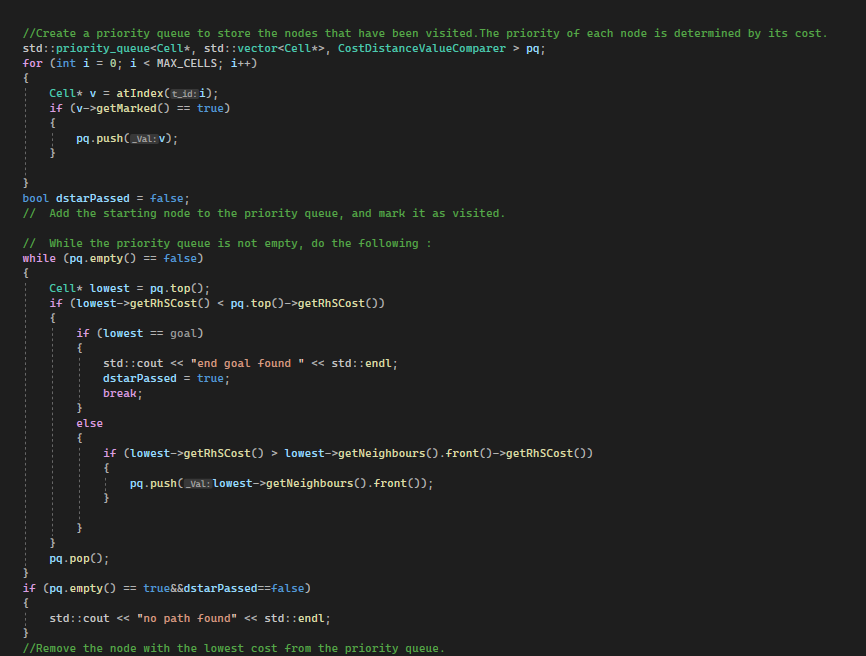
The software will in essence function as a visual pathfinding application. So the user will run the application and see a basic grid they can then adjust the size of the grid as they see fit for example 20x20 , 1000000x1000000. They can then also choose whether to use the A star pathfinding algorithm to traverse this grid or use D star to do so. They can also place down obstacles during runtime onto the grid which will have the algorithms react to them and still find the most suitable path.

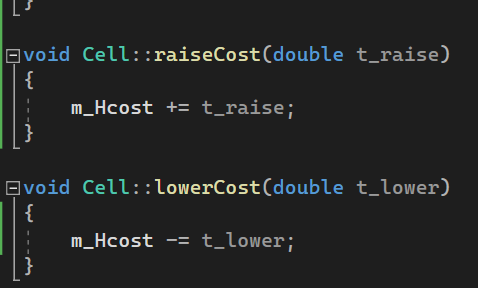
 the basic grid which is able to be changed in colour and size

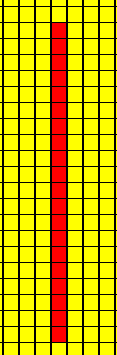
 the Astar algorithm being ran and displayed in the console there will be a visual representation in the application

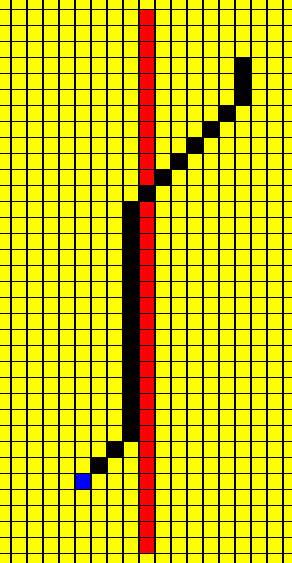
 My Astar function working properly

 the Dstar algorithm being ran and displayed in the console there will be a visual representation in the application

 my currently broken Dstar function

 my raise and lower functions which raise and lower the Hcost of a node (cell in my code)

 an example of basic walls.

 path returned using Astar

***Dstar will be added once fully functioning. Currently incomplete.***

Design and describe how the application will be used: e.g. Navigation sequence in Web based project.

The user will be met with a grid will ask which algorithm they want to use. They can then place obstacles as they wish and as such run the pathfinder. They will then get a string pop up on the screen which will suggest the better algorithm for the scenario either Astar or Dstar.

User interfaces should be specified loosely here (possibly screen shots of prototype user interfaces - you will not be held to using these UIs they are just to assist in explanation of your project functionality).

Include any element that you would feel conveys how you perceive your project will function.

# References

# 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