Computer Games Development

Project Report

Year IV

Quintin Furlong

C00218453

29/04/20

Contents

[**Acknowledgements** 1](#_Toc22543931)

[**Project Abstract** 1](#_Toc22543932)

[**Project Introduction** 2](#_Toc22543933)

[**Background** 2](#_Toc22543934)

[**Project Description** 2](#_Toc22543937)

**Overview**

[**Results and Discussion** 3](#_Toc22543939)

[**Project Review and Conclusions** 3](#_Toc22543940)

[**References** 3](#_Toc22543941)

**Acknowledgements**

I wish to thank the people mentioned below for their assistance in the completion of my project,

**Philip Bourke of IT Carlow** who helped me in narrowing down the specific aim/goal of the project, so it was neither too big of a project for me to do properly nor was it too easy to dedicate an entire project towards.

**Lei Shi of IT Carlow** who advised me to include alternate (standard) path finding algorithms and a way to show how efficient each was, so I could show the importance of my specific algorithm.

**Project Abstract**

The problem is creating an algorithm for multiple agents to get to their own destination without getting in each other's way and increasing travel distance. Path finding algorithm work under the assumption that nothing in the same space will move, so in a situation where multiple agents will be moving through each agents has to redo paths for the unknown new position of other agents before each movement. This will also cause unnecessary movement along soon to be abandoned paths. A solution to this problem is having all the path together and fixing the conflicts before any agents starts down the wrong path.(This paragraph does point out how my algorithm may be better for process time but my project is focused on the efficiency of the paths created not on the efficiency of creating them)

**Project Introduction**

I chose this topic because of my interest of seeing AIs react to one another while still trying to achieve its own specific goal. The algorithm could be used in warehouses with automated workers, where having shorter paths increases the amount of product they can move. It could be used in games to have many NPCs walking past each other in a more natural movement.

Does an algorithm planning out multiple paths in advance create shorter paths then multiple algorithms working in real time?

**Background**

The game “Cities: Skylines” is a city building game where you must build the houses, shops and roads to connect everything together. Part of the game is to build the road network so not only can you get from any building to any other, but also to avoid causing bottle necks which lead traffic jams. The idea of an A.I. car having to take the highway because too many A.I. cars were clogging up the city let to idea of an A.I. algorithm to control multiple agents.

**Project Description**

The project is a grid world with goals, blocks and agents who want to get to their goals and can’t go through blocks. The user control when the agents move and can change the type of pathing they will use.(changing pathing will reset agents). There are numbers to display the number of moves, the pathing being used and an average movement for all agents. There are 5 different path finding algorithms, the first two are simply to show how complex an algorithm needs to be to just get each agent to their goals. The third algorithm has each agent calculate it’s next move and then moving before anything else. This algorithm works by treating the other agents as blocks. The fourth just calculated the full path at the start ignoring other agents, to show the best possible paths. The fifth algorithm and the main focus of the project gets the best paths, finds the conflict and calculates the best alternates paths with no conflicts.

The technical achievement of this project was the algorithm for multiple agents that either matches or does better then a single algorithm for each agent.

I personally have a much better understanding of paths and editing paths to avoid specific locations.

**Overview**

**Results and Discussion**

When the agents each use their own algorithm, they take an average of 16.1 moves to get to their own goals.

When the agents don’t care about overlapping they take an average of 14.6 moves to get to their own goals.

When using the final algorithm that takes all the best paths and fixes the overlap they take an average of 14.8 moves to get to their own goals.

The difference between the first two is to be expected but between the first and third shows the importance of the final algorithm.

**Project Review and Conclusions**

Being able to actually make the final algorithm and not being too process heavy is clearly good. Having other algorithm in the project, helps highlight the usefulness of the algorithm.

If the project was made with the idea of randomisation, it would better show the versatility of the algorithm, possible even allow the user to set up the game during runtime. Even more advanced mechanics like having a first goal(to collect something) and second goal(drop off).

**References**

<http://idm-lab.org/slides/mapf-tutorial.pdf>