

Conclusions

Dijkstra's algorithm is a well used path finding algorithm, however, when searching for a single goal improvements can be made by adding direction to the algorithm known as the A* algorithm. A* performs better on dense graphs where dead ends are limited and improvements to performance can be made on sparse graphs by increasing the influence of the heuristic value but can have negative impacts on the quality of the path generated.

1. Introduction

This project is an investigation on different techniques used to path find within games. Two popular algorithms of Dijkstra[1][2] and A*[3][4] are used and pitted against each other to determine which one performs best in what given environment. Improvements will be applied to A* using a multiplier[5] to see if further improvements could be made. OpenGL is used to give a visual aid and also allows for customisation of the map data.

2. Objectives & Methodology

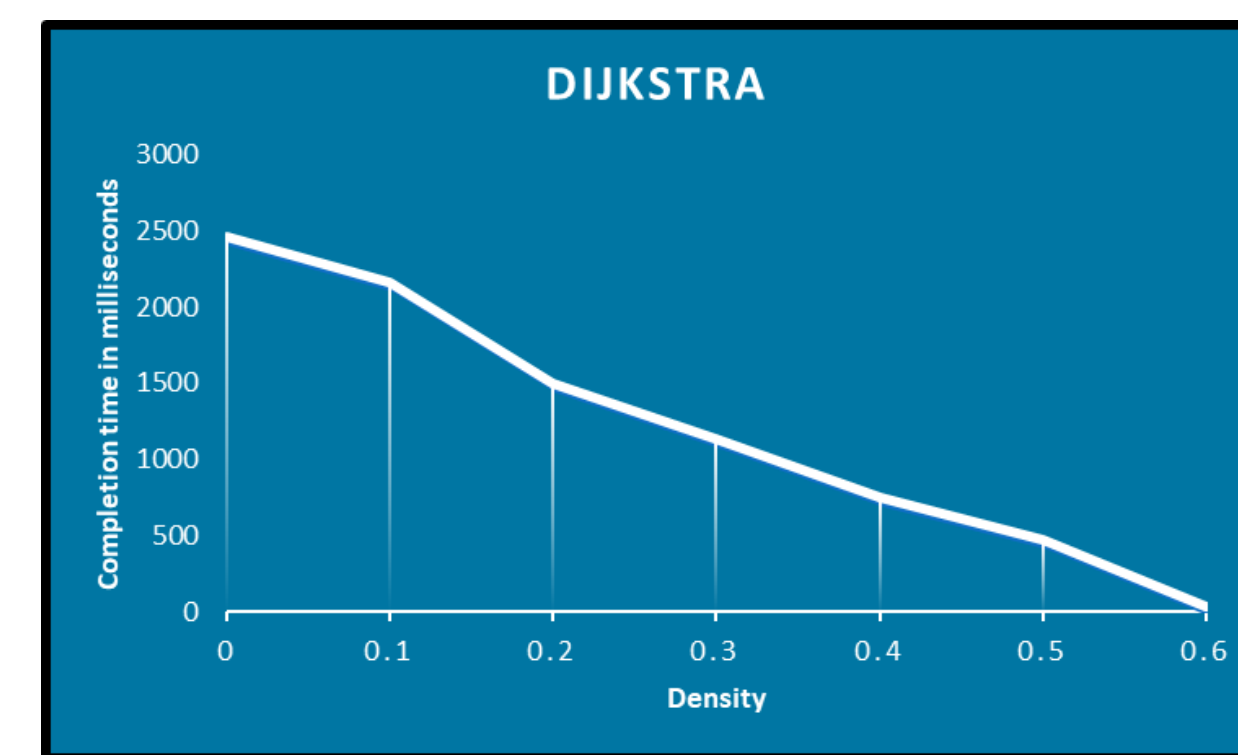
Feature Driven Development was used for the project and tasks were broken down into small chunks making them more manageable. The tasks were

- Investigate and implement the Dijkstra and A* algorithm using C++
- Create a map generator to create areas with different levels of density
- Compare the two algorithms and display relevant graphs showing nodes created, nodes processed and completion times
- Apply a multiplier on A*'s distance heuristic and check for improvements using different types of maps
- Convert from standard console to OpenGL to display output of the completed paths generated by Dijkstra and A*
- Create a map editor that allows for editing start, goal, wall and floor positions and then allow for the algorithms to be run on these

3. Results

3.1 Dijkstra

Dijkstra algorithm was run on different maps ranging from 0 to 0.6 density levels where 0 was an empty map with no obstructions and finished on 0.6 where 60% of the map was covered with obstructions.

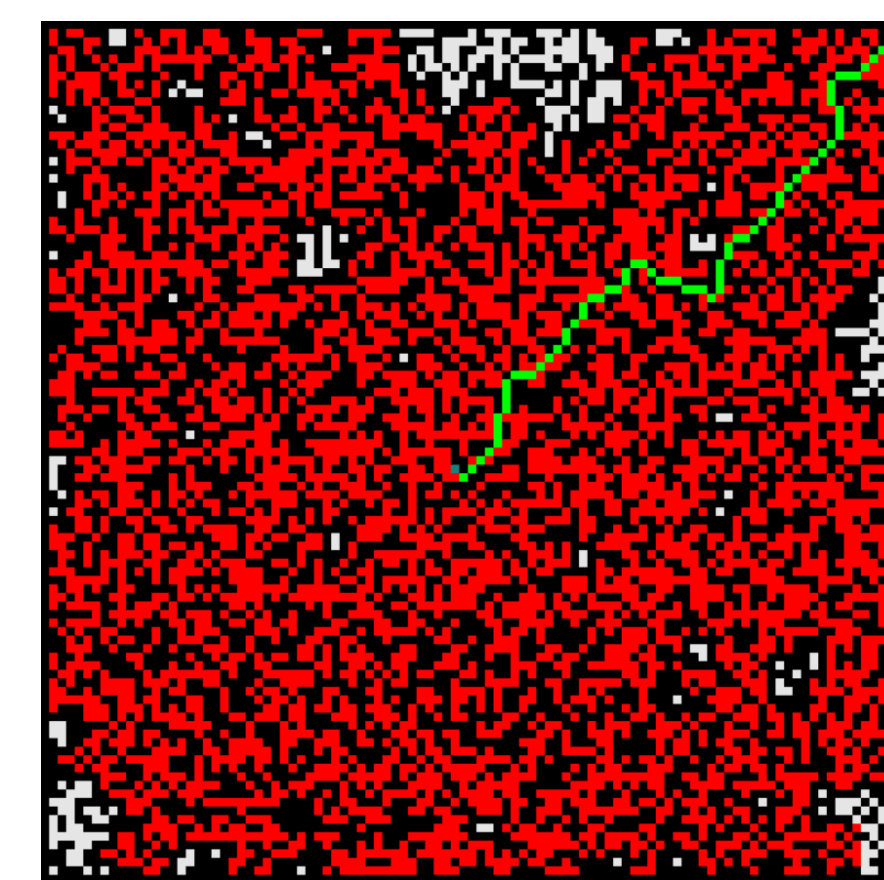


Dijkstra completion time in milliseconds on different densities

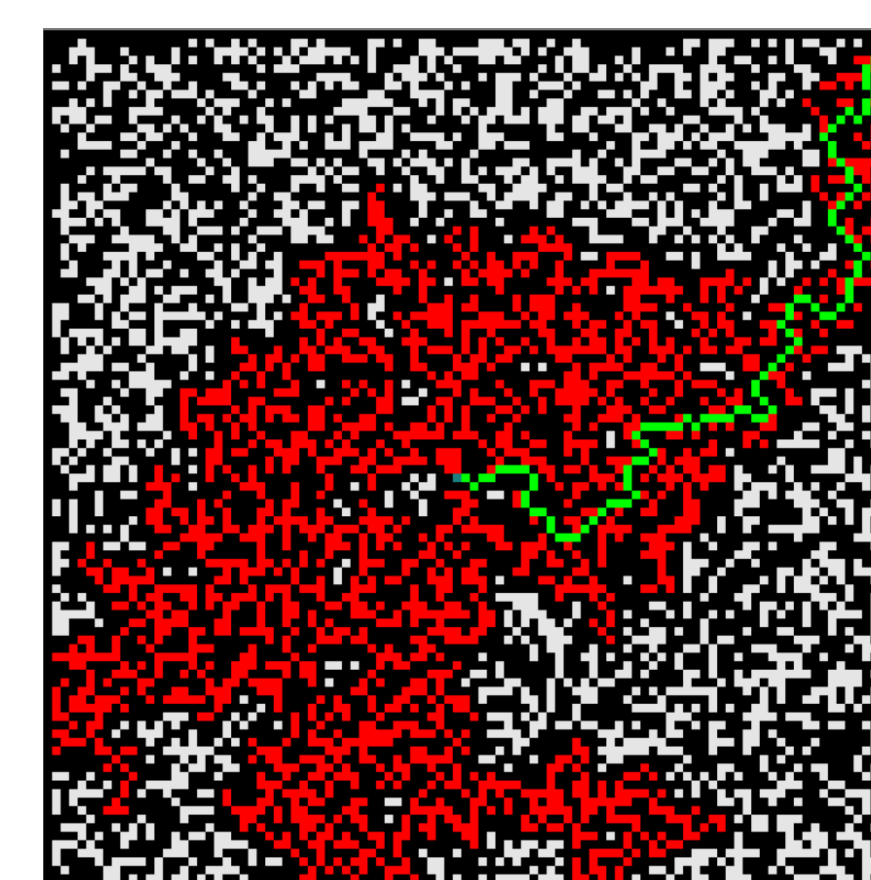
Dijkstra	
Density	Processed Nodes
0	9801
0.1	8811
0.2	7820
0.3	6829
0.4	5815
0.5	4543
0.6	1830

Number of processed nodes for the completion of the algorithm

Dijkstra improved in performance as less nodes were available, however, improved drastically as many areas were blocked off restricting the algorithms movements as shown below.



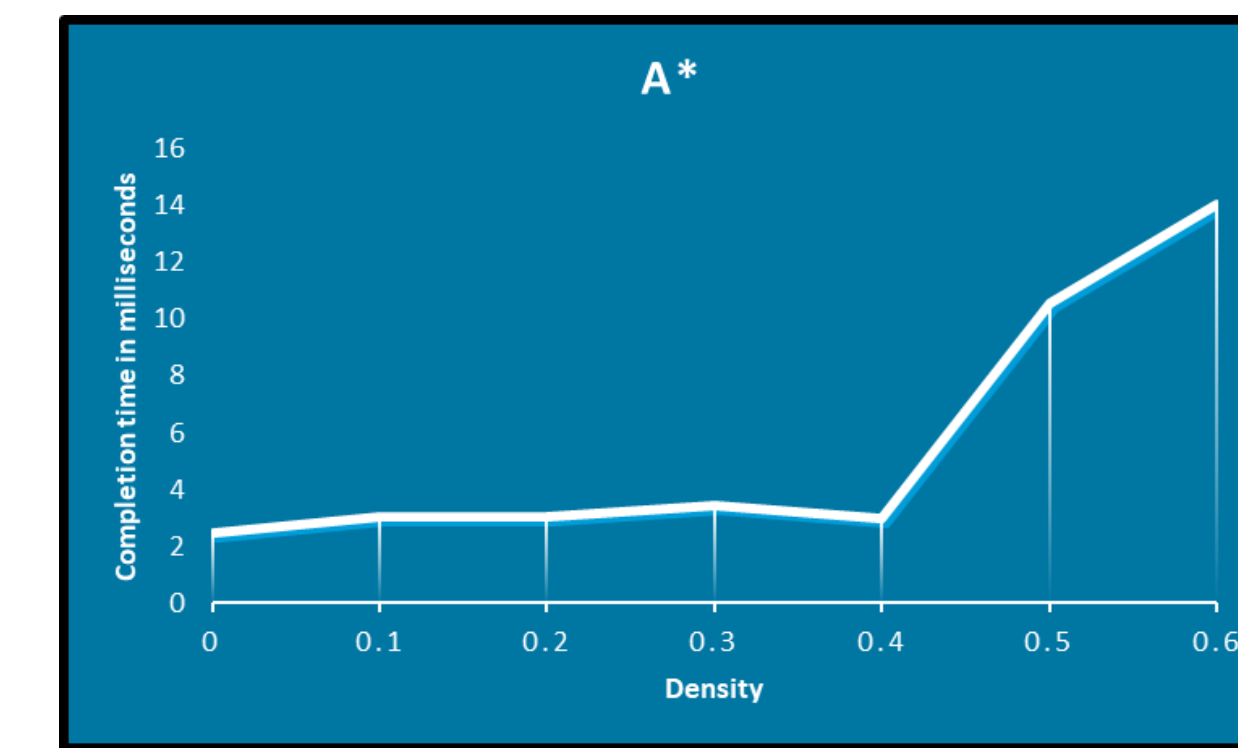
Processed nodes on 0.5 density



Processed nodes on 0.6 density

3.2 A*

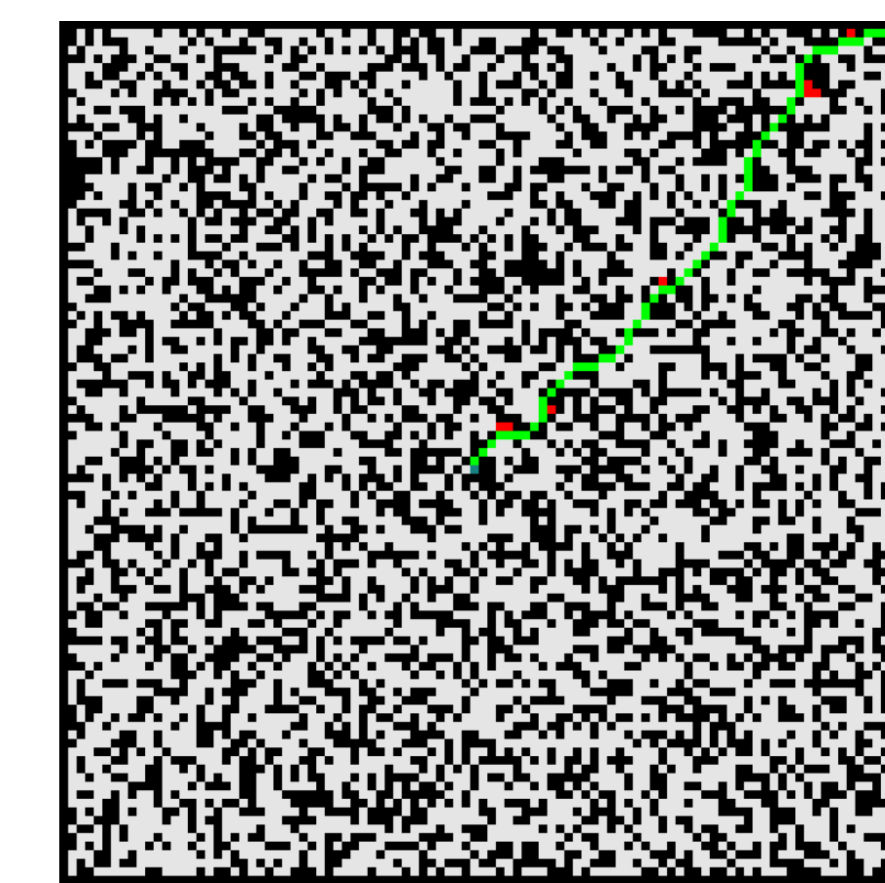
A* made improvements on Dijkstra but performance decreased as the number of nodes available decreased due to being lead down paths where backtracking was required.



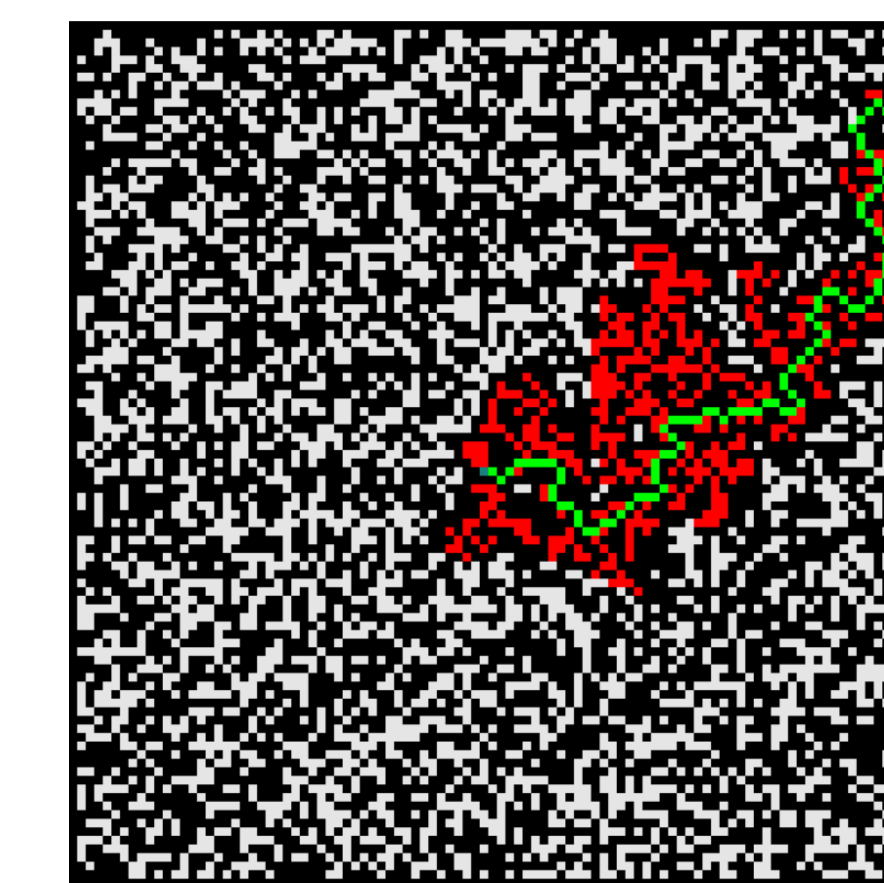
A* completion time in milliseconds on different densities

A*	
Density	Processed Nodes
0	52
0.1	57
0.2	98
0.3	82
0.4	78
0.5	346
0.6	701

Number of processed nodes for the completion of the algorithm



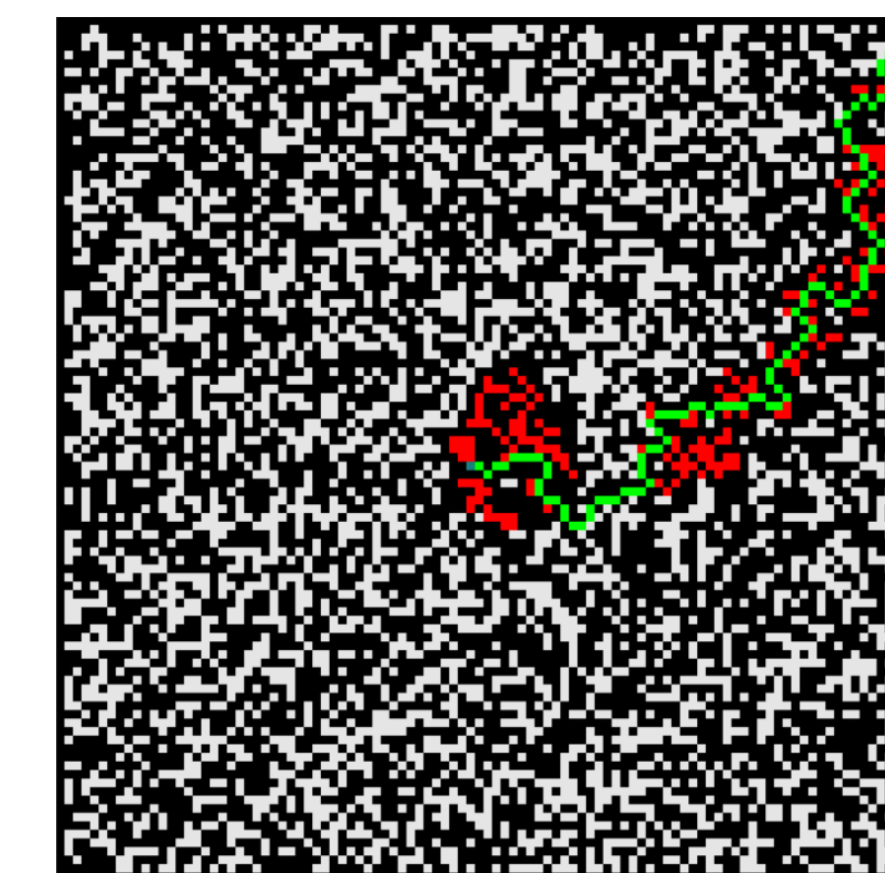
Processed nodes on 0.4 density



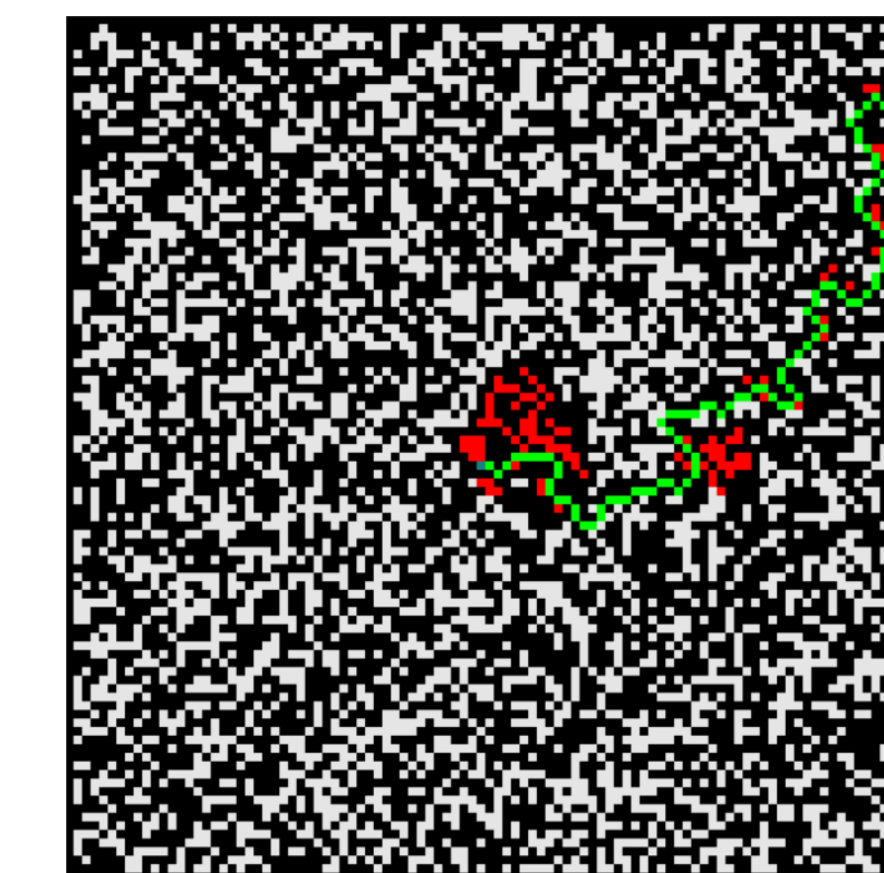
Processed nodes on 0.6 density

A* Distance to goal calculation influence

Nodes that are closer to the goal were given more influence which provided faster results over standard A*, this however, led to a slightly longer path making it less optimal.



Processed nodes on 0.6 density with 1.2x influence multiplier



Processed nodes on 0.6 density with 2.0x influence multiplier

Multiplier	Density		
	0.2	0.4	0.6
1	3ms	2.94ms	14.06ms
1.2	2.27ms	2.32ms	3.55ms
1.4	2.85ms	2.08ms	4.46ms
1.6	3.36ms	2.99ms	7.58ms
1.8	3.3ms	3.66ms	4.55ms
2	1.89ms	4.13ms	4.54ms

Time taken at different influence multipliers

Multiplier	Density		
	Created	Processed	Path
1	1596	701	102
1.2	904	342	103
1.4	863	327	103
1.6	832	305	103
1.8	811	295	105
2	708	293	105

Nodes created and processed with different influence multipliers showing degrading of the path

References

- [1] E. W. Dijkstra, "A Note on Two Problems in Connexion with Graphs," Numerische Mathematik 1, pp. 269 - 271, 1959.
- [2] A. S.-H. Vinther and M. S.-H. Vinther, "Pathfinding in Two-dimensional Worlds," 2015
- [3] I. Millington and J. Funge, Artificial Intelligence For Games, Morgan Kaufmann, 2009.
- [4] M. Buckland, Programming Game AI by Example, United States of America: Wordware Publishing, Inc, 2005.
- [5] S. Rabin, Game AI Pro, Natick: CRC Press, 2013

Further Information

- LinkedIn - linkedin.com/in/andrew-humphreys-9b517711/
- Email – Humperdink01@gmail.com
- Mobile – 07913006279
- Facebook – facebook.com/andrew.humphreys.121

Acknowledgements

I would like to thank Gordon Dickers and Andrew Baker for support throughout the project, also I would like to thank UWTSD for the resources and opportunity to do the project.