Real Time Crowd Simulation

Peter Maloney1

13002995

Supervisor: Dr Simon Scarle

Submission Date 14/04/2016

Module: UFCFS4-30-3 Creative Technologies Project

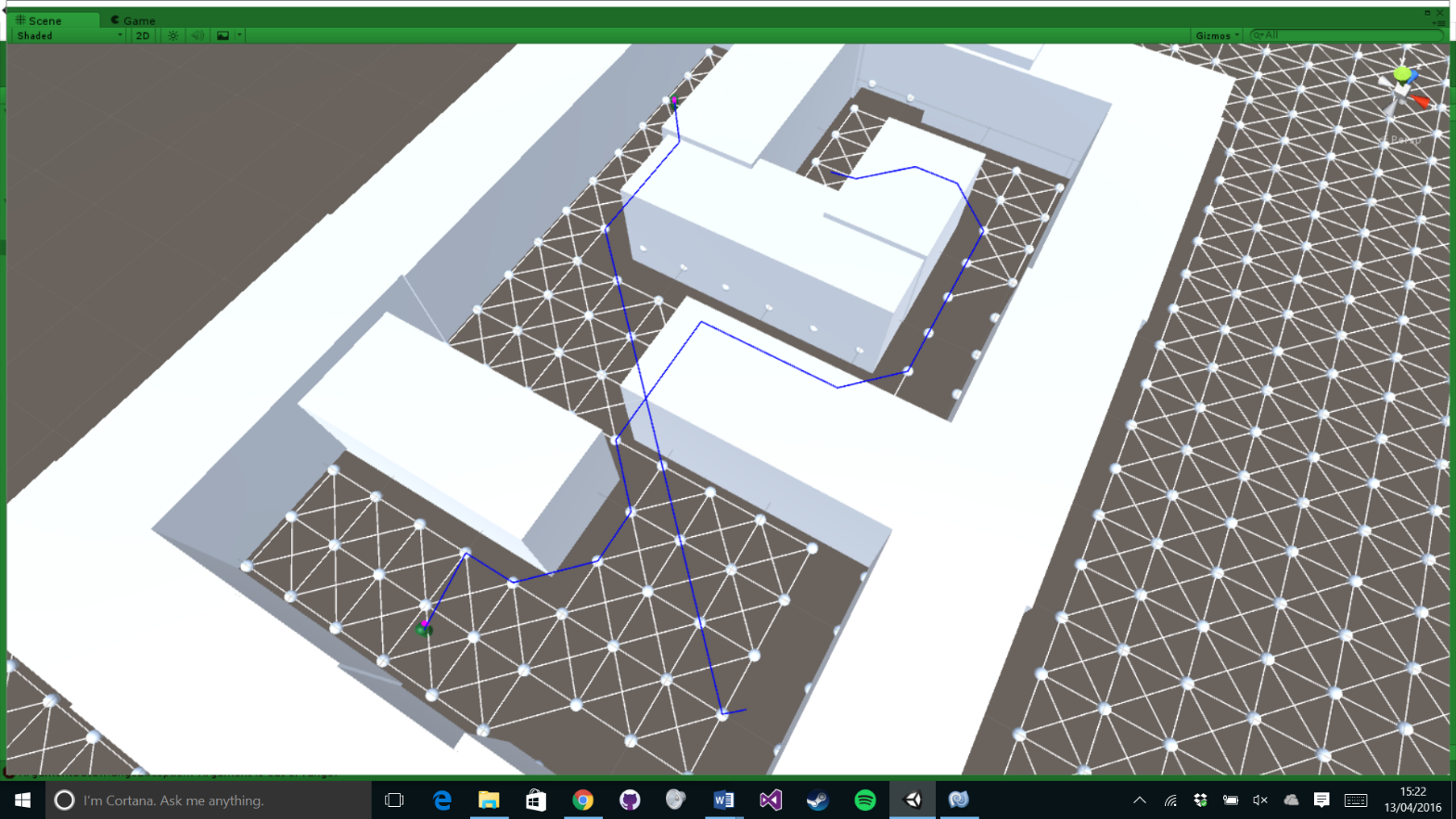
1 Department of Computer Science and Creative Technology.

University of the West of England

Coldharbour Lane

Bristol, UK

[Peter2.maloney@live.uwe.ac.uk](mailto:Peter2.maloney@live.uwe.ac.uk)



1. Summary

The final output of the project in question is a basic system comprising of two main components, which combine to simulate crowd behaviors, both autonomously and guided by user input. The system in question contains much of the functionality that was initially proposed, but lacks a large amount of stability and usability that would be expected.

The first main component of the system is a pathfinding sub-system, which utilises *Dijkstra’s Algorithm* (Dijkstra, 1959)*,* or variations thereof, to calculate the shortest distance route between two points in a grid of user defined nodes. This system itself makes use of components which detect connections between the nodes, to allow large grids to be easily set up for use. This sub-system outputs a list of the positions of the nodes on the shortest path, which is then used by another component, inspired by *Left 4 Dead’s* AI system (The AI Systems of Left 4 Dead), to ‘smooth’ the often jagged paths created due to the nature of the node grid. This component aims to smooth the path by looking ahead to the furthest visible point on the path, and adjusting agent movement to focus on this, cutting out many of the sharp turns of the un-touched paths.

The second sub-system is a collision avoidance system, which aims to produce more realistic behaviors of agents within the system. While avoidance of larger, static objects is dealt with by cutting node connections through said objects, this sub-system deals with avoidance of smaller, dynamics objects, such as other agents. This sub-system is again inspired by *Left 4 Dead’s* AI System (The AI Systems of Left 4 Dead), making use of two colliders in front of, and off to either side of each agent. When either of these colliders detects a collision, it will send a message to the agent, causing it to turn away from the collision, producing, in most cases, smooth and somewhat realistic behaviours.

1. Biography

The developer of this project is a final year Games Technology student at the time of writing, with experience with C++, Unity, and aspects of artificial intelligence as a result of studies. The developer takes a personal interest in video games, and their systems, particularly real time strategy games. By undertaking this project, they hope to develop a system that could be used for future projects, as well as to gain a better understanding of artificial intelligence systems and the potential uses for said systems.

1. How to Access The Project

The entire project can be accessed at: <https://github.com/phatpedro21/CreativeTech>

The project comprises of One Unity project, titled ‘CreativeTechGithub, which will run with Unity Version 5 (older versions may be suitable). The latest version of the project found in the folder titled ‘A\_Presentations’, is made up of Two example scenes:

* The scene titled ‘Sandbox’ is set up to show off all the components of the system, and was the scene used to produce the examples seen throughout this report as well as the demonstration video.
* The scene titled ‘General Showcase’ is intended to show the system being applied in a beautified scenario, and should produce behaviors without any user input, by simply ‘playing’ the scene.

Within either of these scenes, the arrow keys will pan the camera and the mouse wheel with zoom the camera in and out. Left clicking on an agent, will select it and right clicking anywhere with an agent selected will order the agent to move to that point. IT may be necessary to press the ‘G’ key to make agents move, if they do not by default.

There are also a number of options available in the inspector. On any object with the Pathfinding Showcase script these are:

* A check box for ‘Check Distance’, checking this will cause pathfinding to use an A\* approach, unchecking will use Dijkstra’s Algorithm.
* A check box for ‘Yield Yes’, checking this will spread pathfinding over multiple updates, allowing other actions to be carried out while a path is found, unchecking this will perform pathfinding in one frame, which may cause the system to ‘hang’.
* A check box for ‘Show Checked Nodes’, checking this will highlight each node as it is checked by the pathfinding system.

On any object with the Agent Showcase script the available option is :

* A check box for ‘Smooth Paths’, checking this will draw lines showing the actual path taken by agents, that is the ‘smoothed’ path as opposed to the exact path created by the pathfinding system.

1. Introduction

This project set out to produce a system that could be utilised to primarily as a tool for implementing realistic crowd behaviors within video games, specifically RTS (Real Time Strategy) games, but a guiding factor of development to was to ensure the system was as versatile as possible, opening up potential uses to include the creation of CGI crowds in film or building realistic simulations for training, to name a few examples.

Alongside the production of a useful tool, this project also set out to explore the possibilities with regards to realistic simulation in real-time, and potential options for optimization of systems to accommodate this. Another line of investigation that this project necessitated was into what ‘realistic’ crowd behaviors actually are, and whether or not these behaviors could be simplified down and recreated programmatically without losing the appearance of being realistic.

1. Methodologies
   1. Research

The key research questions of this project were :

* What is ‘realistic group behavior’ and of these behaviors have been successfully recreated digitally?
* What are the common requirements for crowd behaviors systems within RTS games?
* What methods of collision avoidance currently exist, and how suitable would they be for the system this project aimed to produce?
* What methods of pathfinding currently exist, and how suitable would the be for the system this project aims to produce ?

To answer these questions, research was carried out using a number of different sources, such as research papers, video games and existing projects and codebases. The main findings from this research were :

* Many of the behaviors observed in crowd situations can be represented mathematically, in many cases this representation is relatively simple.
* Even if isolated, these behaviors can appear visually complex, and imply some sort of intelligence.
* In order to be suitable for use within an RTS game, it would be desirable for a system to be capable of simulating about 400 agents.
* A number of existing collision avoidance systems have been conceived, each with their own positive and negative aspects with regards to the project.
* While there are a number of potential existing pathfinding systems, one in particular, *Dijkstra’s Algorithm* may be the most suitable for this system.
  1. Implementation

Development of the system was a largely iterative process, adding a small component, refining its implementation and then moving on to the next component. This iteration was interspersed with research in order discover possible methods for implementation, or alternatives to planned components when initial ideas were found to not be suitable. This approach was chosen to improve the chance that at least one component could be implemented and presentable at the end of development, rather than many half-finished components that do nothing to present work done. For the most part, this approach was successful, allowing development to fully focus on each component as they were added and meant that any issues faced at any stage could be given full attention, and were only relevant to the current stage of the iteration, saving the time of having to find the potential source of an issue. The aspect of leaving some research of potential options until ends of unsuccessful iterations was also beneficial as experience and finding from work up until that point could guide research, providing more practical results. Despite the positives however, hindsight reveals a number of factors that were problematic with this approach. Firstly, the separation of implementation of components often lead to problems where previous components were not fully designed with future components in mind, meaning time was often lost reengineering work that had already been completed. This problem could be overcome by better structuring the process, planning the order of implementation as well as taking the time to fully consider all of the shared aspects of each possible component. A second issue arose because the process aimed to complete each component to completion before moving on, which became problematic when said implementation became difficult. From experience it was found that it could be hard to decide if and when to admit defeat and attempt another option, and the desire to have something working at the end may have led to some options to be abandoned too soon.

The methodology made solving problems fairly simple, as they were most often isolated, and could be faced with a good degree of freedom, as if any attempt to solve it was not successful, a previous, working iteration could be returned to. Most of the problems faced with this project were technical, and the approach to solving them, was most often trial and error in the short term, trying variations of the existing implementation, and failing this alternative options were explored. Both of these practices would draw from initial research, or exhausting those, require further research. This further research could be a lot more focused than the prior research, as a more specific question was being asked, and lessons learnt while trying to solve the issue could help to build more useful questions.

1. Results
   1. Overview

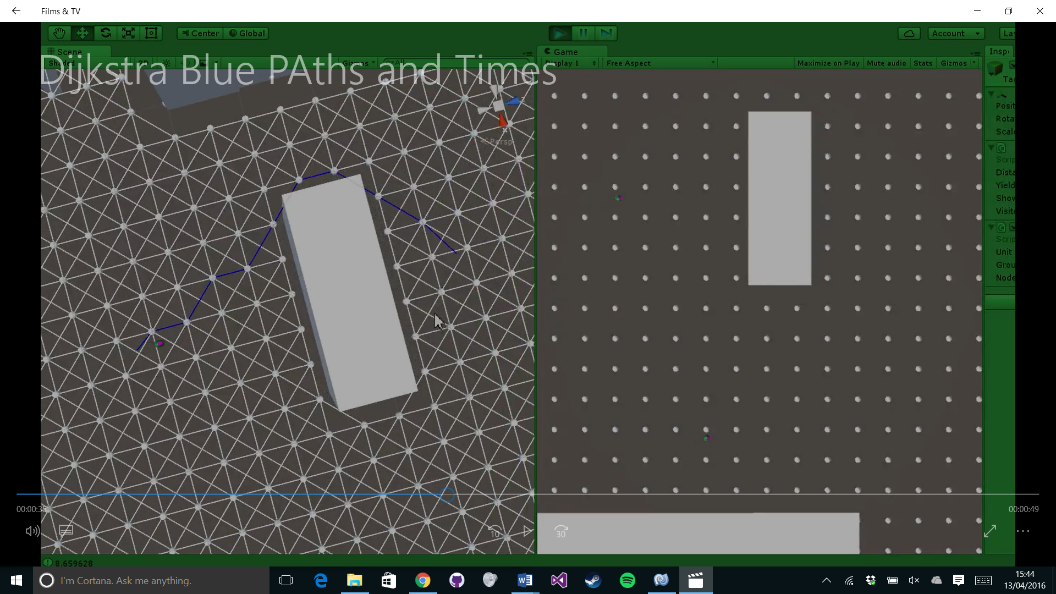
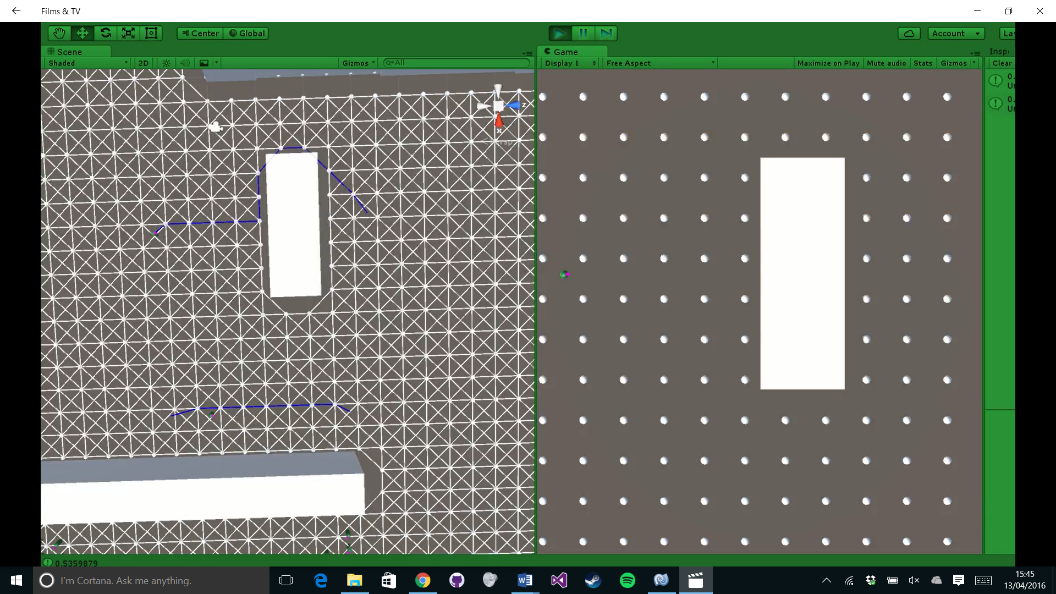
The original aim of this work was the production of a crowd simulation system, primarily for use in RTS games. The output fulfils this aim, however it is somewhat scaled down, and does not realise the full scope of the initial vision. As well as a technical system, over the course of development, the research questions have been answered, and while not all answers have been directly utilised, they have all played a part in the development cycle.

The overall final system combines a pathfinding system and a local collision avoidance system, which produce movement behaviors for agents. Many of these agents navigating in a shared environment then creates a crowd simulation. The final system however, does not contain a number of desired features, such as ‘individuality’ of agents, or wider functionality and usability which would make the system a useful tool for developers.

It should also be noted that while in some circumstances the system is capable of running in real-time, larger environments and some other factors limit its viability in this sense.

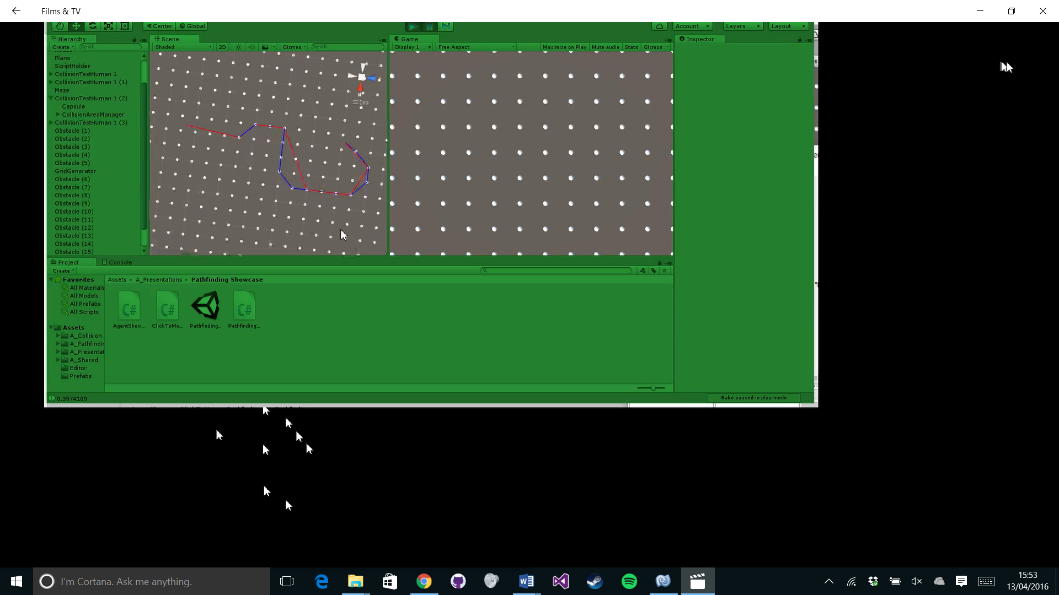
* 1. Pathfinding

The pathfinding component of the final system utililises *Dijkstras Algorithm* to generate paths for agents among a grid of nodes. *Dijkstras Algorithm* was chosen as it will always give the shortest route between nodes assuming edges in the graph (the cost from one node to another) are not negative, making it reliable, but also because the algorithm is the basis for other pathfinding algorithms such as *A\**, providing options for variations in the pathfinding system, which is useful for trying to better optimise the system for real time use.

**Fig. 1.** Examples of the paths built by the system, seen in blue. Left shows path built using Dijkstra’s Algorithm, Right shows path where the system prioritises searching nodes that are closer to the goal.

The current implementation of the pathfinding system consistently find the shortest route and can be used for finding routes between nodes on a pre-defined node grid as well as between nodes added during runtime, which enables RTS style, user defined movement. However, it often compromised the real time capability of the system, especially when producing long paths. This however can be improved somewhat by providing it with heuristics (making it operate more like an A\* system).

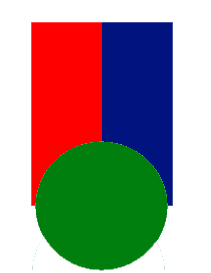
As the paths returned by the pathfinding component are based upon a node system, they are sharp and jagged, and would not produce particularly realistic behaviors should an agent follow then directly. For this reason, another system has been implemented to attempt to smooth these paths out. This system draws inspiration from one of the systems *Left 4 Dead’s AI* use, and looks ahead to the furthest visible node, and uses this as a target, bypassing any redundant nodes in between. This makes paths more direct, and while they are still somewhat robotic, combined with the collision avoidance system, smoother, more human behaviors are produced.



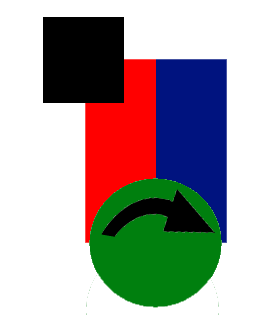
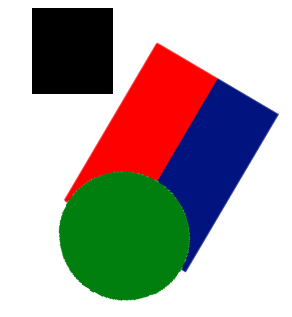
**Fig. 2.** An example of the difference between the path produced by the pathfinding system (blue lines) and the ‘smoothed’ path (red lines).

* 1. Collision Avoidance

The collision avoidance system used in the final iteration is also inspired by *Left 4 Dead’s AI* system, and makes use of two collision detection areas in front of each agent, one top the left, the other to the right. Whenever something colliders with either of these areas, the agent will begin to turn away from the obstacle, and then return to facing in its desired direction once the obstacle is no longer in the way. This is a simple system that is highly effective at avoiding collisions, and produces smooth movement. The behavior produced by this system is slightly realistic due to the removal of collisions and some of the behaviors produced as a result of this however the local rather than global nature of the collision avoidance, and a lack of ‘forward planning’ leaves the results lacking in realism when observed closely.

****

**Fig. 3.1.** Representation of the arrangement of collision avoidance areas (red and blue rectangles), on each agent (green circle)



**Fig. 3.2.** Representation of how collision avoidance systems operates. For example, if an obstacle (black box) collides with the left collision area (red area), the agent will turn to the right, taking it around the obstacle.

The main reason for a collision avoidance component was to deal with avoiding agent on agent collisions, a common problem in real life crowds, and less so on static obstacle avoidance, as the pathfinding system should handle most of that, and it is successful in this regard, removing instances of agent on agent collisions, without causing agents to veer too far off course. This is particularly visible when avoiding head on collisions, as agents will always prioritise turning one direction over the other, avoiding potential incidents of agents both moving the same direction and get stuck in a pattern that takes them further and further away from their desired path. While this component is not very ‘smart’, and does not have any foresight, it is effective at producing some smart looking and realistic behaviors, although it is likely that testing it in more scenarios may lead to some undesirable or problematic results, and an alternative method, such as one making use of Velocity Objects, as were researched, would provide a much more stable and reliable solution.

* 1. Failures of The Final System

While many aspects of the final output of the project are successful and of a satisfactory standard, there are a some of issues with the system which compromise its suitability for use. These issues are:

* The collision avoidance system is not reliable, and often leads to agents straying very far from intended paths, and also, in some cases becoming stuck on an obstacle. Successful implementation of a system similar to *Clear Path* (Guy *et al*, 2009) would be expected to overcome these issues.
* The pathfinding system appears to be unable to calculate more than one route simultaneously, and attempts to do so will result in errors and agents being left without paths. This limits the systems suitability for use within RTS games. While potential reasons for the issue have been speculated, a possible solution could not be found.

1. Discussion
   1. Realisation of Aims

The development of the process has been informative with regards to the question of whether or not what had been set out to be achieved was realistic and achievable. While we already knew that a number of individual systems have been created to simulate different aspects of human locomotive behaviors and interaction of these behaviors in a crowd environment, and that these are often capable of doing so in real time, and that systems for producing similar behaviors within video games also exist, we have come on to find that the combination of these individual systems is in many cases possible, and opens up a number of possibilities for systems of varying degrees of realism and complexity. In this sense, it could be argued that the project aim is realistic, and that the creation of a realistic crowd simulation is not only possible, but approachable in many different ways. However, some of the other aspects of the initial aim have been found to be potentially less achievable. While a number of individual systems have been seen to run in real time in some instances, they do not in others, such as the use of Dijkstra’s Algorithm for pathfinding system becoming less and less viable as the set of nodes used increased. Because of this, the hope that the system being produced could be versatile and used for a range of applications should be deemed less realistic.

* 1. Other Findings

The outcome of the project also suggests that a single complex systems may not be needed to produce some of the more complex behaviors, as a combination of simpler systems can sometimes produce similar behaviors, as also suggested by *Reynolds Boids model* (Reynolds, 2001). This opens up a possible investigation into exactly what is possible using this approach, as while complex systems may produce much more desirable outputs, their complexity may limit accessibility, as was found with our attempts to implement a system using Velocity Objects*,* where a lack of understanding of the mathematics and theory of the system made implementation ultimately not possible, a combination of more easily understandable components, that could produce relatively similar quality outputs would open up said functionality to more developers.

* 1. Potential Application

With regards to the actual system produced, with some refinement it has potential for use in both personal games projects, and as a tool for other games developers, while also providing a good foundation for other specialized systems, as thanks to the iterative development approach and lessons learnt from this, the current system is modular in nature, providing opportunities to mix and match components to suit the needs of a potential variation. With this functionality, systems could be produced for a number of uses, many of them beneficial to society such as simulations within emergency training systems or architecture and urban planning for example, where the use of virtual simulations saves time, money and can allow for better services as many scenarios can be observed quickly and easily.

1. Conclusion

At the end of this project, it has become apparent that the specific expected outcome, to create a versatile, multiuse crowd simulation tool, of the investigation may have been beyond the scope of what was achievable, simply due to the size and variation of needs of possible uses, but an example of a potential system for just one of these possible uses has been created, although it is lacking in polish and depth. Despite being unable to fulfil the expected outcome, a large amount has been learnt about potential development structures for such a task, what kind of similar technology exists and the possibilities and limitations in such a field.

Using the foundation set by this project, work on making the system more efficient and useable as a shareable tool for other developers would open up enticing further investigation, and see that the investigation carried out so far finds real world applications.

1. References

Reynolds, C [2001] Boids Background and Update. Available from: <http://www.red3d.com/cwr/boids/> [Accessed 28 November 2015]

Dijkstra. E [1959] A Note on Two Problems in Connexion with Graphs. Niimerische Mathematik 1, pp. 269-271.

Booth, M [2009] The AI Systems of Left 4 Dead. Available from : <http://www.valvesoftware.com/publications/2009/ai_systems_of_l4d_mike_booth.pdf> [Accessed 13 April 2016]

S. J. Guy, J. Chhugani, C. Kim, N. Satish, M. Lin, D. Manocha, and P. Dubey [2009] ClearPath: Highly Parallel Collision Avoidance for Multi-Agent Simulation.  
Computer Animation, Conference Proceedings, pp. 177-187.

1. Bibliography

Video Explaining Dijkstras Algorithm <https://www.youtube.com/watch?v=WN3Rb9wVYDY>