Daniel Holmes

Supervisor: David White

Second supervisor: Craig Weightman

AI-Controlled City Traffic/Pedestrian Simulation

GDEV60001 Games development project

Contents

[Abstract 2](#_Toc157351258)

[Introduction 2](#_Toc157351259)

[Aims and Objectives 2](#_Toc157351260)

[Literature Review 2](#_Toc157351261)

[Research Methodologies 11](#_Toc157351262)

[Results and Findings 12](#_Toc157351263)

[Discussion and Analysis 12](#_Toc157351264)

[Conclusion 12](#_Toc157351265)

[Recommendations 13](#_Toc157351266)

[References 14](#_Toc157351267)

[Appendices 15](#_Toc157351268)

[Appendix 1 – xxx 15](#_Toc157351269)

# Abstract

***An overview of the project***

This is a summary of the whole report’s contents. Readers may decide whether to read the whole report based on the abstract and therefore it should provide enough information for them to understand what the dissertation is about, including the results of the investigation.

The abstract is written last, even though it is presented at the beginning. It should describe the work that has been carried out, not the work that will be carried out.

# Introduction

This investigation will focus on research into the various methods and AI algorithms used across titles that include a ‘living’ city to create realistic behaviours for agents within these virtual environments. Following this research will be the development of a small-scale city simulation artefact implementing the best methods discovered during research to create AI traffic/pedestrian agents that are able to navigate a city environment.

Through the development of said artefact, improvements to already existing algorithms and AI pathfinding methods will hopefully be discovered and implemented into the artefact. These new discoveries can then, along with the researched methods, can be tested to determine how effective they are at creating a realistic city, further expanding the knowledge-base and methods available for developers wishing to create a ‘living’ city themselves. The research and artefact will also hopefully result in identifying the best methods out of those discovered by others, allowing future developers to identify what methods and algorithms are most effective when creating a simulation of this kind.

# Aims and Objectives

**Aim:**

The research and analysis of methods used in creating an AI-driven city simulation, with said methods then being used to develop an artefact displaying how effective the researched methods are and how they can be improved.

**Objectives:**

* What algorithms can be used to achieve a realistic city simulation?
* What actions should an agent be capable of performing?
* How do agents interact with other agents in a realistic way?
* What makes a city simulation believable?
* Can the researched best AI pathfinding method be implemented successfully to produce adequate agent behaviour?
* How can the success of agents within the simulation/the simulation as a whole be measured?

# Literature Review

***How are other people doing it?***

AI and Video Games have an interlocking history that dates back to the inception of Video Games as a type of software, understandably meaning that over the years a plethora of different AI creation methods have been theorised and practically implemented into commercially released games of all different genres. Believable vehicle AI is no exception, leading there to be a large selection of games to research and methods of creating said AI available for use depending on the kind of game you want to make and the behavioural complexity you want the AI to have. As my research is focusing on vehicle and pedestrian AI in the context of a city builder/simulation game, I have ensured that one of my sources details methods used by a game that creates a simulated city using vehicle AI.

In Chapter 17, titled “Fast Cars, Big City – The AI of Driver San Franscisco, of the 3rd edition of Game AI Pro, authors Chris Jenner and Sergio Ocio Barriales go into detail discussing Driver San Francisco’s unique AI system the game needed due to its main defining mechanic. Driver San Francisco is a game focused on high-octane car chases both involving the player and independent of them, taking place in a simulated virtual San Francisco. The player is able to freely disembody themselves from the vehicle they are driving and can take control of any vehicle that they can see at will. This meant that vehicle AI needed to both be capable of being a part of the independent background city simulation that made up the game world the player inhabited, said AI had to be capable of taking over control of whatever vehicle the player leaves, regardless of the state of that vehicle, and make the vehicle behave as expected for whatever kind of vehicle it is. For example, if the player leaves a police car in favour of a different vehicle whilst they are wanted by the police, the believable behaviour for that now AI controlled police car would be to chase whatever vehicle the player now occupies. Whereas if the car the player leaves is a civilian, that civilian would likely want to drive away and get out of danger away from the wanted criminal the player currently is.

This system was coined “Active Life AI”, and is one of two AI solutions present in Driver San Francisco, as classic pathfinding solutions (such as A\* pathfinding) were inadequate for handling the dynamically changing environment created by the player changing what vehicle they are controlling. The other AI solution present is the “Civilian Traffic AI”, which handled the movement of other vehicles along a pre-determined spline in a loop, with no interaction between two splines being present. The active life agents needed to be capable of blending in with the other civilian traffic agents, as well as racing or chasing other vehicles controlled by the player or other active life agents. Utilising a simplified version of the Active Life AI method will likely be necessary when creating the artefact for this dissertation, as if the behaviour of agents is to be believable, they need to be capable of some degree of autonomous decision making separate from simply pathfinding the optimal route to their destination and following it. An agent being able to evaluate the state of the road ahead of them and act accordingly, recalculating the best path they should take or prioritising specific road types, will create more realistic behaviour across the city simulation as a whole. Active Life AI may even allow for realistic agent response to collisions/accidents, or even further emergent behaviours. The developers also didn’t want the AI to cheat when competing against the player, so they made sure that the agents had to control vehicles using the same driving model the player was using whilst being able to compete with the best human players. This is part of the AI system that can be discarded when designing the system that will be used in the artefact.

The chapter also goes into depth in describing the methods used for calculating vehicle paths, as well as describing how roads are represented to the agents. Each agent has a vehicle path, detailing the predicted movement of that vehicle over the next few seconds, this path being updated approximately every second. Upon updating, new sections are added to the end of the path before the vehicle has travelled, making the path from the perspective of the agent endless. A system similar to this would likely be useful to implement into the artefact vehicle AI solution, most likely on a less extreme scale, meaning that the path of the vehicle is updated a designated points either in time or on the path the vehicle is taking. This would allow for the path of the vehicle to adapt to the changing road conditions around them, as long as the path took into consideration the changing environment around and ahead of the vehicle when being added to. Concerning how the path is calculated, the writers describe how path generation method was determined by what was controlling a vehicle at the time, the player, the Civilian Traffic AI, or the Active Life AI. For player-controlled vehicles, physics and dead reckoning are used in path generation. The traffic splines are used for path generation when a vehicle is controlled by Civilian Traffic AI. Active Life path generation is the most interesting and complex of the three, utilising a three-tiered level of detail (LOD) method to determine how a path is generated depending on the range of the vehicle.

The lowest LOD generated their paths using only route information, with the route being defined by a list of roads they can take to get from point A to B. Roads themselves are defined as splines connected together with junction pieces. The path for these low LOD vehicles is a portion of the route’s spline with an offset applied to simulate the vehicle driving in a specific lane, the offset being updated along with the path every second. The next LOD utilises mid-level paths to generate vehicle paths. First using an approximation of a good path for a vehicle, then using route information along with extra details such as lane information, preferring to maintain the vehicles position in their current lane if possible, and finally some rough dynamic object avoidance and speed limit data. The highest LOD is used for vehicles within the vicinity of the player, specifically those within the view of the player camera, with said vehicles using full three-tier path generation (route finding, mid-level path planning and low-level path optimisation, these methods are discussed later in the paper). A system of this complexity will be unnecessary for the goals of the artefact, however parts of it can potentially be used for general agent pathing, especially when it comes to defining routes using specific roads.

How roads are defines in Driver San Francisco is also done using a method that will likely be considered for implementation in the artefact, most likely with modifications. The roads within the game are defined using splines, with one spline representing one road. Each spline has a start and end extremity with a list of roads attached to each extremity describing each potential road that could be used after exiting this current road. The extremities also contain cross section information detailing the number of lanes, width of the lanes and lane types at that extremity. The example given in the chapter displays a spline with two “with traffic” and two “oncoming traffic” lanes, detailing the types of lanes that were possible with this spline system. The example also shows that the splines have defined sidewalks, which if implemented into the artefact could mean that vehicles and pedestrians can use the same spline system for route planning.

Utilising how roads are defined, Active Life agents in would begin their pathfinding process by selecting the roads it plans to use to reach its destination, the goal of this stage being to generate a list of connected roads (a route) that can be used. The method of generating this route would depend on the goal/behaviour of the vehicle the agent controls. The first method, and the one that is most likely to be used within the artefact, is a traditional A\* search of the road network when the destination of the agent is known. The second method is a dynamic, adaptive route generator that was used when the objective was to escape a pursuer. The second method can be entirely disregarded, as there will more than likely be no situation in which the destination of an agent is not known in the artefact. Route planning also had multiple other factors that might influence the route an agent would decide to take, including driver personality, ensuring the route is physically possible for the vehicle to undertake considering the simulated vehicle physics system each vehicle uses, and other negligible factors, however none of these are relevant to the objectives the artefact needs to achieve.

The next stage of route finding involved path optimisation, a stage which the designers named “Mid-Level Path Planning”, would define the lanes the agent should use on a specific road in the route. This was achieved using a series of nodes placed at regular intervals across a spline within the route, with the nodes each defining a lane on that spline. The mid-level path was then generated by looking ahead of the vehicle far enough for the generated path to be validly used for the next few seconds, usually looking 100 metres ahead. The nodes were placed in such a way to create a grid in front of the vehicle, so that a path could be generated through these nodes for the vehicle to use. Each node also had a dynamic numerical value assigned to it when a vehicle was using them to calculate a path, so that the most desirable node path could be found based on the total sum of the nodes evaluated. Some examples of conditions that would affect the score a node received include whether a dynamic obstacle (e.g. a vehicle) was near, giving a penalty to that node if that were the case. Nodes containing static obstacles (e.g. buildings) could not be used at all. Vehicles would also have a preference for driving in a straight line, again affecting the score the nodes would be given. The chapter features example diagrams that visualise this system, aiding in understanding how this system is used. A system that is this complex will likely be unnecessary for the artefact, however some kind of lane definition and look-ahead method will need to be implemented to make sure vehicles do not collide with each other or other objects/pedestrians around them, and that vehicles use the lane that is most optimal for them when on a specific road. Further path refining was also undertaken to ensure that the path generated was physically possible for the vehicle to drive, with this stage of path refinement being coined “Low-Level Path Optimiser” by Driver San Francisco’s developers. This system, however, will not be relevant to the artefact, as the vehicle agent’s present in said artefact will not be using a vehicle physics simulation model, likely having no physics simulation of any kind.

Looking more broadly into methodologies used to create AI controlled vehicles in video games, Chapter 39 of the 1st edition of Game AI Pro, titled “Representing and Driving a Race Track for AI Controlled Vehicles”, discusses and details methods with which a developer could create a race track that can be driven by an AI controlled racer. Authors Simon Tomlinson and Nic Melder start by giving a general overview as to what features may be present in an AI driven racing system, emphasising the importance of how the track itself is represented to the AI driver and that, regardless of diversity and amount of data present in one game’s system, there will always be a representation of the physical layout of a racetrack. Usually, there will also be a racing line bundled together with the data that represents the tracks physical layout, giving a primary fastest route around the track. Said racing line is also influenced by factors other than the track layout, as depending on the genre of racing game there may be power-ups/speed boosts present that result in a faster lap than just pure driving. They also discuss how real-time evaluation may be needed if the vehicles have enough varying characteristics that a racing line cannot be applied universally to all vehicles in the game, but ideally both would be present.

Going into more depth, they discuss how to represent the physical layout of the track in data form, that the AI can then interpret for navigation. At a minimum, the representation of the track must define the boundaries of the racing area, so that the AI knows where it can drive. Within this boundary, the track can be marked using a series of nodes, placed along the nominal track centre. Each node has an associated perpendicular and normal vector to define both the width and orientation of the track at that node. Nodes are arranged in either an array or bi-directional linked list to form a series of quad segments, which is represented visually in figure 39.1 found within the chapter. This method means that the width of the track never needs to be symmetrical, and as long as the path defined using the nodes is reasonably smooth, the AI driver will have no problem driving the track. As seen in figure 39.1, drivable areas that aren’t necessarily on the tarmac (e.g. runoff areas) are still represented using this method of physical track representation, using three values on either side of the track to define the main boundary of the track, the extended runoff boundary, and the hard width which represents the solid walls and boundaries the track is within.

Comparing the two methods of representing a physical road to an AI driver discussed in the 1st edition Game AI Pro, Chapter 39, and the 3rd edition of Game AI Pro, Chapter 17, the methods described in Chapter 17 are significantly more applicable to this research than the road layout methods used in Chapter 39. Despite Chapter 39 providing a more complex representation of the road to be driven by the AI, allowing for changing road widths, definitions of areas not necessarily within the width of the main tarmac of the road but are still drivable, for city simulation purposes the level of detail provided in Chapter 39’s solution is unnecessary. Most likely, the width of the roads found in a city simulation will be uniform, having perhaps 3 or 4 different road widths present that need to be represented to the AI, so having these hard coded into specific road types will be adequate. Road width will also most likely be described to an AI driver in terms of number of lanes present on a road, rather than a numerical width. Chapter 39’s solution does not provide any method for defining road lanes to the AI, understandably so considering this method is supposed to represent a racetrack, whereas the solution described in Chapter 17 does. Since the width of the lanes themselves will likely be uniform across all roads in the simulation, describing the width of each lane, let alone the road itself to the AI is unnecessary. The more important data to give to an AI would be the mid-point of each lane, which Chapter 17’s method does using both the road splines and node grid system.

Continuing on with reviewing what is discussed in Chapter 39, the authors discuss the potential for having the track representation be generated automatically rather than by being manually defined by a developer. To do this, the polygons that make up the surface of the track would need to be processed to produce the layout representing data set. This would certainly save development time, especially when considering the number of tracks that may be present in any particular racing game. However, the authors do not recommend using this kind of automation, as it can lead to inaccurate data when handling areas like street junctions and track runoff, meaning that a developer would then have to manually fix these sections of the track themselves, potentially taking more time than if they had manually defined the track themselves. The authors conclude that having developers place the node themselves, with the assistance of a node placing tool, is the better solution for producing track layout data sets. Last minute edits to the nodes, as well as adjusting nodes to change AI behaviour, will also be significantly easier if the data set was already made manually by a developer.

They then go on to discuss how open-ended tracks and branching paths within tracks could be represented, as it is not always the case that a track can be represented using a singular set of nodes that form a closed loop. Concerning branching paths in a track, representing that the track branches some distance before the track physically does branch is important, especially if vehicles are travelling at high speeds, to give adequate time for the most appropriate racing line to be generated. They then go on to discuss solutions for if a track is not a closed loop, rather being a point-to-point race. Null links on the first and last node will occur if no additions are made to the node system to prevent this, as problems will occur for an AI driver if they happen to drive beyond these nodes. To counteract this, the authors explain that there should always be a run-out distance for a track that facilitates the AI driver either slowing to a stop or driving out of sight to be deleted when crossing the finish line.

Track node spacing is something that also needs to be considered when creating the track layout data set, as the authors describe an understandable trade-off between more nodes with shorter spacing between them leading to a more accurate representation of the track, and the development time needed to place said nodes and assign data to them. Not having a set distance between each node and allowing for variable spacing between nodes is described to be a good compromised solution, with node spacing being shorter on highly curved sections and larger on long straights, despite the extra overhead this would produce from the AI now needing to sum track distances segment by segment rather than simply taking the number of nodes present and multiplying the total by a known constant distance between nodes. There may also be issues that arise from sudden changes in length between segments, which would need to be mitigates through a more gradual change in segment length when transitioning from long straights to curved track sections. If the road representation methods were to be utilised in a city simulation, node spacing would likely be more uniform, as most roads will be simple straight roads set out in a grid, meaning that node spacing can change gradually from shorter spacings near road junctions to larger spacing when driving on the main portion of a road. However, if a city layout were to go beyond this and ended up with a large number of heavily curved, winding roads (perhaps this city is built in a hilly/mountainous area so these winding roads are necessary), then automating node placement would be required simply due to the time that would be required for a developer to manually place the nodes themselves on roads across an entire city.

Calculating track data is the next topic discussed in this chapter, with the authors detailing that there are two types of track data that need to be considered. The first is the position of the AI on the track, which is derived from the AI vehicle’s position in the current frame relative to the nearest sets of nodes. The vehicles distance from said nodes can return a wholly accurate point to represent the vehicles current position. This data is coined by the developers as “track registration” and can be used not only as a reference point as to the position of the vehicle, but can be further processed to determine other information, such as the vehicles’ orientation relative to the track at that frame. The second type of track data is information related to driving hints, such as the maximum viable speed for an upcoming track section. This data can be baked into the track itself, however the information shown in these hints is likely to change depending on what vehicle is being driven, so it is more practical to calculate this in real time. Instead, information such as the radius of curvature of the track, the racing line and other data that can be used in calculating hint data can be baked into the track itself, as these data sets will remain uniform regardless of the vehicle being driven. For a city simulation, only one of these two types of data would be needed, as track hints are irrelevant if none of the vehicles are ever directly controlled by the player. Even in the instance that vehicles can be controlled by the player, unless it is in the context of the race there is no need for the kind of driving hints that the authors describe. The position of a vehicle at a given frame, however, would be very useful and is a feature that should be considered when determining what methods are going to be used in artefact development. This data can be used for a multitude of things, including checking if a vehicle is within a lane, checking vehicle progress along their current route, and checking for vehicle collisions with other objects.

The next section goes into more detail about how “track registration” can be performed, a process which involved calculating three individual elements, the nearest consecutive pair of nodes that define the link segment the vehicle is currently within, the distance the vehicle is along that segment, and the perpendicular distance from a reference line. A visual representation of this process can be seen in figure 39.2, along with the formula that can be used to find the distance the vehicle currently is along this link segment. Considering that it is unlikely that the node system detailed in this chapter will be used in the artefact, there is little need to further evaluate exactly how track registration is performed. However, if any parts of the track layout methods detailed in this chapter are used within the artefact, they will be discussed later in this paper.

Much like the previous section, the next two section detail how to calculate and represent the curvature of the track, describing the process and giving equations to use for finding track curvature. Figure 39.3 specifically gives a visual representation as to what parts of the track correspond to which elements of the radius equation present in the same figure. Again, much like the previous section, there is little need for further evaluation of how the track node representation system works, as it is unlikely to be used in the artefact. Particularly in the case of calculating road curvature, if splines are to be used to represent roads, then any curvature will already be known, and real-time calculation will not be needed. More than likely, most roads within the city simulation are going to be straight, so will have no curve to be calculated.

Following on from the first half of the paper, the second half contains sections all relating to how the AI will drive the track, discussing frame-by-frame driving following the main racing line, as well as how the AI can handle situations in which it cannot follow the best racing line, being prepared to make appropriate adjustments depending on the changing race environment. The authors first go into detail about the use of a look-ahead runner for following the racing line. Specifically, this runner is used for vehicle steering, using the racing line as a guide, as it is better for a vehicle to aim for a point ahead of itself rather than using the racing line directly for steering. If the racing line were to be used directly, the developers found that the AI would continually make small corrections, resulting in the vehicle weaving left and right across the racing line. In the worst case, this weaving would become noticeable to the player, and inevitably leading to the vehicle spinning out of control. Steering should instead be done using the angle between the current direction the vehicle is facing and the vector between the centre of the vehicle and runner aiming point. The developers found that this would smooth out vehicle steering, however this could also lead to strange behaviour in it own right. For example, if a vehicle had a large enough look-ahead distance and took a sharp corner, the AI would cut across the inside of that corner. Look-ahead distance should therefore be calculated depending on both track curvature and the current speed of the vehicle, with the authors stating that finding the exact formula is a process of trial and error. The runner is updated every frame, with any corrections being made concerning its distance from the vehicle. Implementing this kind of look-ahead system will most likely be needed within the artefact, specifically so that vehicles avoid colliding into other vehicles/pedestrians as well as ensuring that they can slow down to a stop before reaching any junctions. In a worst-case scenario, it will be better for a vehicle to stop completely if they are about to hit something or reach a junction even if they would realistically be unable to slow down in time to stop, however a look-ahead runner theoretically would prevent this from happening.

The next two section discuss cornering speeds, braking distance and wall avoidance. However, as discussed at the end of the previous paragraph, the look-ahead runner will likely be used in the artefact for handling when an agent needs to brake for collision avoidance and handling junctions. Wall collisions will theoretically be impossible, as vehicles will have defined drivable road sections using the spline method discussed in the previous literature source, so as long as these splines are defined correctly, there should be no way in which a vehicle can collide with a building. Cornering speeds are entirely almost entirely irrelevant to creating a city simulation, as the look-ahead runner and road speed limits will likely control the speed at which a vehicle will move, and no physics vehicle simulation will be involved with vehicle movement.

Broadening the scope of sources even further away from vehicle AI, looking at AI pathfinding techniques present in video games of genres that aren’t racing games or games that involve simulated cities, Johan Hagelback’s paper “Hybrid Pathfinding in StarCraft” discusses the methods used to develop a pathfinding modification to StarCraft. The objective of this modification being to improve overall pathfinding behaviour for all the units that you control, and by doing so improve the overall gameplay experience. Having responsive and effective pathfinding is especially important for a game like StarCraft, an RTS in which every second lost due to a unit having poor pathfinding gives your enemies an edge over you, potentially stacking up to such an extent that you might lose the game simply due to the poor pathfinding of your units. In the context of the artefact that will be developed out of the findings of this literature review, some of the techniques discussed in this paper may potentially be useful for pedestrian pathfinding.

Hagelback starts by describing that RTS games are usually divided into two categories, macro- and micromanagement, with StarCraft falling into the micromanagement category, focusing on the management of individual units, tactical unit placement, target priority etc. Pathfinding in micromanagement RTS’s is usually handled using traditional algorithms, such as A\*, which can be effective for calculating a path between two positions in a reasonably short time, however Hagelback states that A\* does not work well in dynamically changing worlds, which RTS games tend to have, particularly if parts of the environment can be destroyed or changed in some way. Within the time taken to calculate a path using A\*, that path may now be invalid as a result of changing unit position or map geometry changes, meaning modifications need to be made to A\* so that it can handle these changes effectively. There is also no consideration made by A\* as to tactical unit placement, simply producing the shortest valid path to a goal point. To remedy this, the paper proposes a hybrid approach to unit pathfinding involving using both A\* and the potential fields method depending on the state of the environment around a unit. This system would have units use A\* for pathfinding if no enemy units or buildings are in sight range, swapping to potential fields when units are engaged in combat. Hagelback explains that this solution would avoid the problem of local optima when using potential fields, which is when units can become stuck in complex terrain, by swapping to A\* when outside of combat, whilst also gaining the benefits that potential fields provide for unit positioning in combat situations. The paper also aims to evaluate if the potential fields can be replaced with a system based on a flocking algorithm, specifically Boids flocking algorithm. Hagelback references an open-source RTS game called Glest which has used a flocking algorithm effectively, as well as citing a source that details how a flocking algorithm was used in conjunction with a Bayesian model for unit control in StarCraft, providing proof that a hybrid pathfinding approach has worked for commercially released games.

This kind of system should be considered for implementation within the artefact, specifically when it comes to developing pedestrian AI. As the aim is to create believable pedestrian behaviour, then having some form of flocking of individuals whilst the pathfinding is handled using a separate method will likely succeed in creating the effect of groups of pedestrians moving together towards a goal location. Treating a group of pedestrians as a single cohesive AI pedestrian, as is the case for units in RTS games with a squad of individuals being treated as a single unit, will likely produce this effect, along with singular pedestrians who are not a part of a group. If pedestrians end up using the same navigation system as vehicles, then that will be the other component of this theorised hybrid navigation system.

Returning to the paper, the next section goes into more depth as to how the hybrid navigation system works, going into more depth about the conditions in which an AI would swap from using A\* to using Boids. Hagelback explains that the Boids algorithm would be slightly modified to ensure that units maintain a distance from the enemy close to the maximum range of their own weapons, whilst keeping the group together. This makes sure that units engage an enemy from the most optimal range, only approaching as close as is necessary and potentially out-ranging the enemy they are attacking. Figure 1 and 2 feature diagram that represents the desired behaviour from this method, with the arrows next to the individual members of a unit in figure 2 representing how individuals will try to spread out to stand at the maximum range of their weapons. Figure 3 then shows how this looks when implemented into StarCraft, showing the individual members of a unit standing spread across their maximum effective range. Following on from this, Hagelback discusses how the alternate system that uses potential fields would work, explaining that potential fields work by assigning positive values, negative values, or 0 to areas within the field. Positive values would attract the agent, negative values repel then, and areas with a value of 0 have no influence. Agents therefore move towards the area with the highest positive value. Figures 4 and 5 within the paper display diagrams to visually represent how this would work, once again accompanied by an example of potential fiends being used in StarCraft itself, this being figure 6.

Following on from the previous section, there is a thorough explanation as to how Boids could be implemented into an RTS, with Hagelback explaining that this Boids implementation features six rules, cohesion, alignment, goal, separation of own agents, separation from enemy units, separation from terrain. Each rule explanation is accompanied by a pseudocode example, which will be very useful if Boids ends up being implemented into the artefact. Every rule is also explained to have a “dominance” value which dictates which of the rules take priority in influencing AI behaviour when more than one rule is currently being applied.

The first rule, cohesion, is what is used to keep the individuals that make up a unit together, as Hagelback explains that each individual moves towards the average position of all other members of a unit. This rule is described to be not very dominant, making other rules like maintaining distance from an enemy take priority. The rule following on from this, alignment, also has a low dominance, and is the rule that keeps all individuals in a unit moving approximately in the same direction, specifically moving towards the average heading/direction of the other individuals. The third rule, goal, describes how each unit is assigned a specific goal depending on orders given by either a player of AI controlled enemy, with an example goal being to attack the closest enemy building, and each individual moves towards the goal of the unit. Once again, this rule has a low dominance score. The fourth rune, separation of own agents, ensures that individuals will avoid colliding with other individuals who are in the same unit as them, with individuals striving to keep a short distance between each other. A detection limit value is used to dictate the radius around an individual in which this rule is applied, this value can be adjusted depending on whether individuals are supposed to be closer together or further apart. The fifth rule, separation from enemy units, is the rule with the highest dominance value, and controls individuals maintaining optimal enemy engagement distance according to the range of their weapons. This rules detection limit changes depending on if this unit of the enemy has weapons with the highest range, ensuring units do not rush into enemy shooting range whilst being unable to fire back. The sixth and final rule, separation from terrain, simply ensures that units will not collide with impassable terrain, with this rule not being applied to flying units as there is no terrain in StarCraft that can block flying unit navigation.

Hagelback then details a number of experiments undertaken to determine the effectiveness of the two hybrid pathfinding methods described in this paper when compared with each other, as well as when compared with non-hybrid pathfinding. The results of these experiments show hybrid solutions unsurprisingly outperforming non-hybrid solutions, as well as affirming Hagelback’s assumption that potential fields would have a slightly higher win ratio than Boids. To achieve this higher win rate however, potential fields was seen to have a longer execution time, with the difference between execution times for potential fields and Boids being well above a factor of 100. As a result of this, Hagelback concludes that a potential fields solution would be unsuitable for StarCraft, due to the number of decisions an AI enemy would need to make in the short time available each frame, preventing the AI enemy from running at its highest speeds when using potential fields. This could be resolved in the future with further optimisations of potential fields, however if potential fields were to be applied for pedestrian AI in the artefact, then the downsides of execution time would be irrelevant. Hagelback also posits that potential fields have the advantage over Boids when it comes to its support for representing surrounding in game objects of all different shapes and sizes, due to the nature of being able to attract or repel AI agents to and from different objects in the environment. Hagelback concludes that, for RTS games, Boids can produce results that are at least as effective as a potential fields solution, whilst requiring significantly less hardware resources to do so. As stated before, the increased execution time found when using potential fields would be irrelevant if it were to be used for the pedestrian AI in the artefact, not to mention the benefits it would provide in preventing pedestrians from moving to close to objects in the environment around them, such as buildings.

Concluding on this literature review, there are evidently a large number of different AI pathfinding methodologies available for use in games of all different genres, with this literature sample size only providing a glimpse into what could be used to create effective AI pathfinding in a game. For the purposes of this research, however, there is plenty of information in these three papers to be used to create an effective and realistic city simulation that features both competent vehicle and pedestrian AI. For general navigation of both vehicles and pedestrians, a spline and node system similar to what is described in Chapter 17 of the third edition of Game AI Pro will likely be the most effective at achieving the objectives of this research. Both pedestrians and vehicles will have their own separate spline networks that can be used to reach their desired destination, with these splines being divided down into lanes that agents will occupy depending on direction of travel. The road representation system detailed in Chapter 39 of the first edition of Game AI Pro is by no means a bad method of representing a racetrack to AI driver, but the key problem lies in the fact that said method is designed for racetracks and not regular roads, and therefore contains features that are simply not needed for representing road layouts in a city simulation. The look-ahead runner method for AI navigation, however, will likely be included in some form in the artefact, to ensure that agents are aware of obstacles ahead of them and can slow down in time to avoid collision. Some form of flocking may also be implemented if groups of pedestrians are to be implemented into artefact, however pedestrian groups are not explicitly stated as an objective of this research so it will depend on how successful the artefact is as to whether they are included. In the case that they are, the methods discussed in the “Hybrid Pathfinding in StarCraft” paper will most likely be used, specifically the Boids algorithm described in that paper with less rules, as there will be no rules related to combatting enemies needed for the pedestrian groups.

# Research Methodologies

***How will you carry out your investigation?***

This section is important because if you undertake inappropriate methodology your results and findings will be disputed. The reader needs to know what you did to find out information so they can make a judgement about the suitability of your methodology.

In this section, you state what you have done to achieve your aims, what you did to find the information you need, and, why you did it.

The methodology section can include.

* Research paradigm used, in other words, the type of research you used and why.
* Sample Strategy - if you are using one you should provide a full explanation of who you used in your sample and why.
* Materials and equipment used.

Justify your decisions by referencing back to best practice.

# Results and Findings

***What have you found out?***

Sometimes this section can be merged with discussion and analysis

It tells the reader what you have found out from your investigation. It is objective; there is no interpretation of results in this section (that comes in the discussion). It objectively states the findings of your research. If you have done primary research this is where you present your findings. You may include tables and graphs, but also need to explain the results in words. Any raw data should be included as an appendix.

# Discussion and Analysis

***How has the project gone?***

This covers the interpretation of the findings, evaluation of the significance of the findings and a general discussion of the investigation. What do your findings mean? In this section you should consider questions such as:

* What has your investigation shown?
* Did it achieve its objectives?
* What theory/literature does it support or contradict?
* What are the most plausible explanations of your findings?
* Are there any possible criticisms of the investigation?

The discussion should also:

* Build on the material in the introduction and literature review
* Evaluate the adequacy of your methodology
* Suggest design features that may have affected the results
* Include whether the results would be different under different conditions

# Conclusion

***What conclusions have been reached?***

What has your investigation led you to conclude?

A conclusion:

* Demonstrates that you have achieved what you set out to do
* It provides the reader with a sense of closure on the topic

It might be worth going back to the aims and objectives or your introduction and making sure your conclusion is in line with what you said you would be doing.

# Recommendations

***What would you do in the future?***

Use your findings and analysis to make recommendations. You may recommend that further investigation is undertaken if you realise that there were gaps in your methodology or anomalies in your findings. Alternatively, you may advise that some actions be considered.

# References

Make sure references are given correctly. See Staffordshire University [Refzone](https://libguides.staffs.ac.uk/refzone/harvard) for more information.

We are using Harvard Referencing.

**Referencing guides**

<https://www.citethemrightonline.com/category-list?docid=CTRHarvard>

<https://libguides.staffs.ac.uk/ld.php?content_id=31264350>

<https://libguides.staffs.ac.uk/ld.php?content_id=9572296>

You may want to use Mendeley for your references

<https://www.mendeley.com/>

# Appendices

Appendices is information referred to in the main document. It is not included in the word count.

Do not put results here: only the raw data should be presented in an appendix. Other materials that may be included in an appendix includes, for example, blank questionnaires, copy of written tests used.

Remember do not include anything in an appendix that has not been referred to in the text.

## Appendix 1 – xxx