Joshua Anderson

Innovative Navigation Artificial Intelligence for Motor Racing Games

This is a digitised version of a dissertation submitted to the University of Bedfordshire.

It is available to view only.

This item is subject to copyright.

INNOVATIVE NAVIGATION ARTIFICIAL INTELLIGENCE FOR MOTOR RACING GAMES

By

Joshua Anderson

A thesis submitted to the University of Bedfordshire, in fulfilment of the requirements for the degree of Masters of Science by Research

May 2016

Abstract

Motor racing games are pushing the boundaries of realism and player experience. Artificial Intelligence (AI) allows developers to create believable opponents. By getting their AI to follow a racing line that is similar to that taken by real racing drivers, developers are able to create a sense that the AI racers are trained drivers. This paper identifies two methods used in the field: the sector based system and the sensor based system. The sector based approach offers two or more pre-determined lines for the AI to follow, with added logic allowing the AI to judge when to switch between lines. The sensor method is able to guide AI vehicles around tracks with sensors, offering more possible behaviours and lines. After implementation, the strengths and weaknesses of both methods are realised. The planning and development of a hybrid system was based on these findings. The resulting system is able to produce a more believable line for the AI. With the setting up process of a race track the sector method taking a long time, exploration into tool development is conducted to reduce the process. The subsequent tool reduced the time needed to set up a track, providing results similar to the old method.

Acknowledgements

I would like to acknowledge my Director of Studies Rob Manton for the advice and guidance given, as well as keeping me on task throughout the project. Without this advice, this thesis wouldn’t be in the stage it is today. Acknowledgement is also due to Hong Qing Yu (Harry) for being my second supervisor, and providing feedback this thesis.

Acknowledgement goes to my parents Neil and Doreen Anderson, for the outstanding support shown to me, especially during the most difficult of times. Without the encouragement shown by these individuals, the motivation needed to complete this paper would have been lost, and ultimately, never completed.

I would also like to acknowledge Zachary Cohen, Lionel David, Grigor Todorov and Yi Zu (Joey) for the support given to me during times where it was needed the most. Extra acknowledgement is given to Zachary and Joey who volunteered to take part in a testing process of the project, with the results produced were critical was evaluation of a specific part of research.

Finally, I would like to acknowledge the University of Bedfordshire for funding this degree through the Jubilee Scholarship, and for allowing me to continue on from my previous three years of studying. Without this, this research project would not have been possible.

Table of Contents

List of Figures

Chapter 1 – Introduction

1.1 – The Problem

Racing games are allowing developers to push the boundaries of realism in computer games. Artificial Intelligence (AI) is one area that can affect the perception of realism. Creating believable agents for these games can be a time consuming process, which can be due to the set up process of the implemented system or the behaviours that the developers intend to produce. This thesis will look to make the racing line of the AI in these games more believable by using two methods in areas which eliminate any negative effects of the other, while offering a set-up process that takes less time.

1.2 – Aim and Objectives

The aim of this research project is to discover the benefits of a hybrid system which over existing methods. The believability of the line taken by the AI will be determined by the line shape, the speeds taken by the AI, and total lap time. In order to achieve this aim, the following research questions and objectives need to be completed:

- To research into the different methods of AI production and types.
- To analyse different methods of creating Racing AI.
- To determine the requirements of the AI in racing games.
- To review the artificial intelligence in existing racing games/simulators, in a historical context.
- To explore the characteristics of the different methods available.
- To implement one of the identified methods, along with elements of another into the artefact to produce a hybrid system.
- To produce behaviours in the AI in which are influenced based on the racing conditions, which would make the behaviours more realistic.
- To evaluate the produced system, analysing the results with those from the previously implemented systems.

- To discover the advantages of using a hybrid system over standard methods.

1.3 – Motivation for Research

Motor racing games are at the fore front of computer games technology, with developers being able to implement realistic physics and car handling, allowing for a more realistic experience. However, another part of giving the players a realistic experience is to supply them opponents the player believe are real humans. This can be achieved by offering either a multiplayer game mode to play against other players, or by developing AI for single player.

Non-player characters (NPCs) in racing games "typically consist of a controller designed to follow a target racing line" (Loiacono, 2012), and are usually designed and implemented by the development team. For some games, such as *Gran Turismo 6* and *Forza Motorsport 5,* which have an extensive selection of tracks, this process can be time consuming due to the process each track has to undergo for the AI to be able to navigate around.

Heading into this research, I intend to achieve one aim; to produce a racing line capable of giving a great level of accuracy in following a line that is believable to the player, in a relatively short amount of set-up time. The main reason for this research task was to reduce the amount of development time developers needed to spend in this area, instead allowing them to use the time saved into other areas of game development.

1. 4 – Research Question

To explore the advantages of using a hybrid method to control the navigation of an AI driver around a race track, comparing to discovered base methods.

1.5 – Adopted Methodologies

Due to the aim of the project being to explore the advantages of a hybrid method, this research project followed the qualitative methodology. This was due to the data collection needing to collect samples from specific groups of people (based on play experience, explained in section 3.1), rather than collecting data from a general consensus. This insures that the data used within this thesis is strictly within the three different groups, as well as providing the same amount of data for each skill level between the players.

Chapter 2 – Literature Review

2.1 - Introduction

This section sets out to summarise existing work related to the racing AI. All areas of racing games are covered, from current research areas, types of AI used, and key definitions which will be used throughout the thesis.

2.2 - Introduction to Artificial Intelligence

"AI in a computer game controls all of the entities or agents in the game that has the ability to react to the player or otherwise provide an unpredictable challenge for the player" (Rouse, 2005). As an important part of the single player component of games, AI needs to provide an interesting challenge for players. AI can cover almost any aspect of the game, from NPCs to the way the game processes the rules of its world.

2.2.1 – Pretending to be human

Driving towards achieving AI that feels human-like is a goal that developers push for. This is due to the rise in popularity of online multiplayer games, offering unpredictability with human opponents, as well as social aspects. As claimed by Rouse (2005), it is this unpredictable nature of human opposition that is the main reason why multiplayer games are fun.

Due to this rise in popularity, it is possible for game developers to assume that single player games need smarter and unpredictable opponents. This rise has also seen a decline on the amount of people in which experience the whole single player component. Snow (2011) interviewed Activision contractor, Keith Fuller, who stated that "90% of players who start your game will never see the end of it unless they watch a clip on YouTube." Although this stat doesn't rely solely on the AI of a game, or on the developers, as suggested by Jones (2011), they can ensure that the game and AI is interesting enough to keep the player engaged.

2.2.2 – Importance of AI outside of Single-player

Even though the increase in popularity of multiplayer games can be attributed to the decrease in single player interest, developers still consider producing single player components. One example of this is the game *Quantum Rush*. Originally designed as an online multiplayer racing game, the developers of *Quantum Rush*, GameArt Studio, considered producing a single player version, which can be seen in Figure 2.1. According to Develop (n.d.), these thoughts came from lag issues the game experienced during beta stages.

Figure 2.1: Screenshot of Quantum Rush: Champions. (Ryan, 2015)

AI also has an important role in multiplayer games, as it is incorporated into almost every online multiplayer game. One example is the *Call of Duty* franchise by Activision. Within the online portion, players can unlock an in match reward known as a kill streak. Some of these rewards are controlled by AI, eliminating as many opponents as possible.

Some games offer computer opponents in competitive multiplayer games. These opponents will use AI to make decisions as a player. This AI can serve multiple purposes, with the first being to provide more players. This is useful for games which are best played with a full lobby, or if a minimum number of players is required for the game to begin.

A second purpose is to control a body inside the game when not being used by the player. This AI will behave differently to how the player can normally interact with the world; however, they will behave like the player would when they manually control the body. This AI will only be activated if a player is not controlling the body, increasing the number of competitive components, such as a titan in *Titanfall*, which can be seen in Figure 2.2 being controlled by a player.

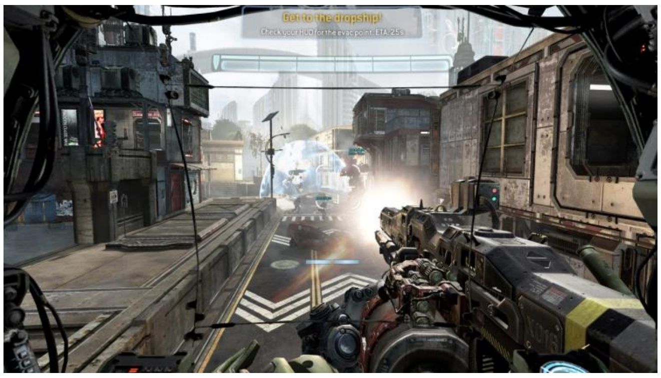


Figure 2.2: Screenshot of Titanfall. (Martin, 2014)

A third use of AI is to offer practise to the player by using ‘bots.’ These AI will offer difficulty levels to match the player’s skill level. This allows players to evaluate routes of each map, and to practise different tactics. This will improve their skill levels, and potentially allowing them to play better against human players. *Call of Duty: Black Ops* by Treyarch (2010) features a mode called Combat Training, allowing users to compete against a number of bots on a multiplayer map, acting as opponents and team-mates for the player.

2.3 – Background to Motorsport

Motorsport comes in many varieties. Despite these variations, the main principle remains the same: complete the race distance in the shortest amount of time possible. The main type of motorsport is track racing, with the majority of the racing series’ using this course type.

2.3.1 – Racing Line

The racing line is a definition used in motorsport to describe the line drivers follow around a track to maximise their performance. This imaginary line, as described by Northern Motorsport (2012), provides the best route for getting around a circuit, allowing it to be “driven in the fastest possible time.” As drivers constantly use this line to get round the course, the line will have increased grip. Northern Motorsport (2012) explains that this is “due to the rubber build up [from the tyres onto the track surface.]”

2.3.2 – Cornering and Apexes

The corners are one of the places on the track where a driver can reduce the time taken to complete a lap. A behaviour which drivers do on approach to a corner is to move to the outside of the track, getting as wide as possible. This reduces the sharpness of the angle they have to get the car to turn. As a benefit, the driver is able to take corners at a slightly higher speed, which can reduce lap times. When the drivers turn into a corner, they do it from a position which allows them to hit a point known as an a*pex*. Northern Motorsport (2012) defines an apex as "the middle point of the inside line around a corner at which drivers aim their cars." Although drivers don't always hit this point, missing it can make their lap times longer, something which could add up to costing them a position or two. Figure 2.3 shows a diagram of the racing line going through a corner, with the apex marked to show the point of the track the car should clip on its way through the turn.

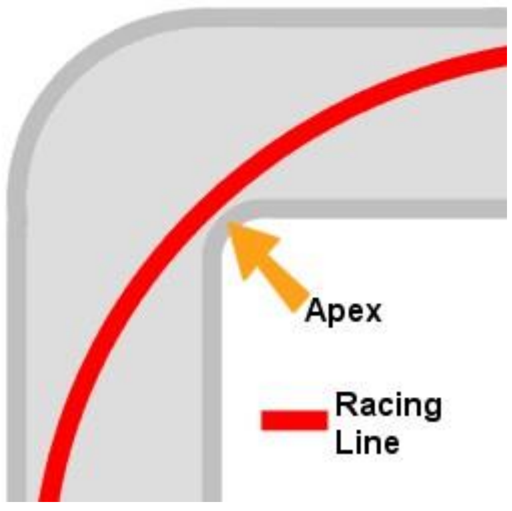


Figure 2.3: Diagram of the racing line through a corner, along with its Apex. (Driving Experiences, 2012)

2.3.3 – Other Important Terminology

Alongside the use of the two main terms defined above, this thesis will be using the following keywords in which are commonly used in motorsport:

Downforce – A force that pushes down on the car, "pushing the vehicle to the road" (Formula1-Dictionary.net, n.d.). The more downforce applied, the greater speeds the car can take corners.

Oversteer – A resulting behaviour where the car experiences too much turn around a corner, with the rear of the car attempts to turn more than the front.

Retirement – The competitor is unable to continue racing due to an issue beyond their control (e.g. mechanical failure, terminal crash damage) resulting in them having to pull out of the race.

Understeer – Opposite to oversteer, as the car experiences too little turn around a corner, which originates from the front end of the car not producing enough turn. Figure 2.4 gives shows a comparison of oversteer and understeer.

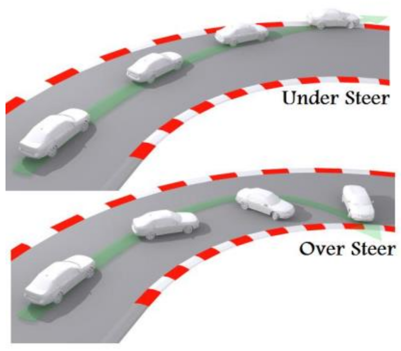


Figure 2.4: Diagram of Oversteer and Understeer. (Smith, 2014)

2.4 - Importance of Artificial Intelligence in Motor Racing Games

Just like most genres, AI is important in racing games. With the most common AI type associated in racing games being the controller for opponents, there is another type of AI controlling the racing conditions.

2.4.1 – Track AI

The first type of AI associated with racing games gets the cars to move around the track. Track AI is usually a state-based system, using different states to represent the ways in which each vehicle can be present on the track (Schwab, 2009). These states can be used to determine what behaviours the AI should perform and what action it should do. One example given by Schwab (2009) is labelled "wrong way.” While in this state, it’s easy to predict how the AI would behave. As the AI races round a set track, it needs to do it in the correct direction. It would use built-in logic to find the correct direction, and get itself facing and moving in that direction.

Schwab (2009) also notes how most games “use a combination of physics and optimal lines of travel (...) that mimic the invisible lines of travel that humans use when they race on tracks and roads” in order to navigate the AI. The invisible lines being referred to here is the racing line, which is important for the AI to follow.

2.4.2 – Race Control AI

Race control AI coordinates the racing conditions of the race as it progresses. This can be anything from setting the flags in certain sections, weather changes and to the race start/finish procedure. An example of this AI in action can be described through the game *F1 2014* by Codemasters (2014).

At the start of a race, players (along with any AI) are placed into grid slots which are determined by qualifying or by random. The race start sequence then commences. During this period, no cars are able to move. Once this sequence is completed, the cars are free to accelerate towards the first corner and the race officially begins. Figure 2.5 shows the start procedure in an online multiplayer race in *F1 2014*.

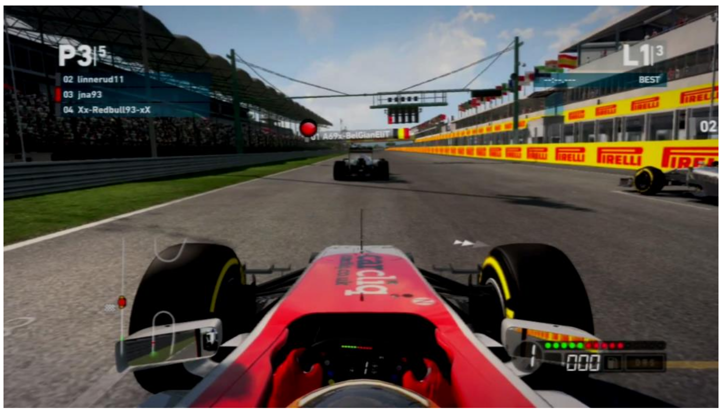


Figure 2.5: Screenshot of the start procedure in F1 2014. (Codemasters, 2014.)

Although this type of AI doesn't directly control the NPC's movement or decision making, it is a very vital part of this genre. Without this AI, racing games would possibly not be as efficient at making the racing experience as real and fair as possible. Since this AI controls the race, it needs to be present in all racing games, whether it be simulation, arcade, single player or multiplayer.

2.5 – Current Research into Motor racing games

Racing games are among the "most popular game genre[s] as they can provide a realistic driving experience in a vivid scenario." (Loiacono, 2012). Because of this popularity, researchers are intrigued in this particular genre, and have been able to identify many areas that would benefit from further research. This section aims to highlight areas which have recently been researched into. Ashley Westgate of Eutechnyx stated that "Motorsports continue to push their boundaries, so as racing game developer we need to do the same" (Batchelor, 2014)

2.5.1 – Player perception and interaction of the game world

One interesting area of research is how the player is able to perceive and interact with the game world. Bateman, et al. (2011) looked into this area, primarily at how the camera position and input controller used can affect the players performance. What makes this area an interestimg topic to delve into is that racing games tend to offer the player a variety of ways to play these games, from different camera options, to choice of controller. Figure 2.6 demonstrates an advanced set up that serious players would use for playing racing games.

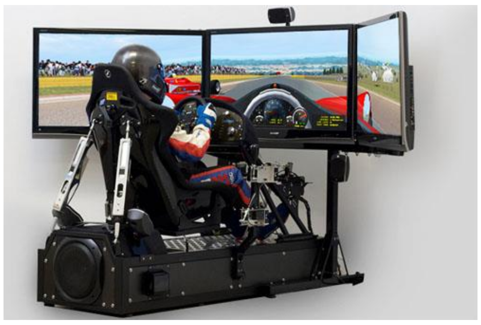


Figure 2.6: An advanced sim-racing set-up. (Paradigm Shift Driver Development, 2015)

Some results presented by Bateman, et al are quite surprising. One result was based on the input method. Despite mimicking what the player would find inside of a car, the study discovered that the steering wheel was not the best device in terms of performance, but rather the thumb stick of a gamepad. Another detail that was picked up was how well the players performed while using the different camera positions. The overall performance showed that first and third person cameras on the car were more beneficial to the player as opposed to using an overhead view. This included a zoomed out version of the overhead view, revealing more of the track ahead.

The main reason why developers tend to add the compatibility of devices such as a steering wheel and (more recently) virtual reality (VR) headsets is to give a great sense of realism and immersion. One racing game that currently offers VR headset support for the *Oculus Rift* is *Project CARS* by Slightly Mad Studios (2015), which can be seen in Figure 2.7.

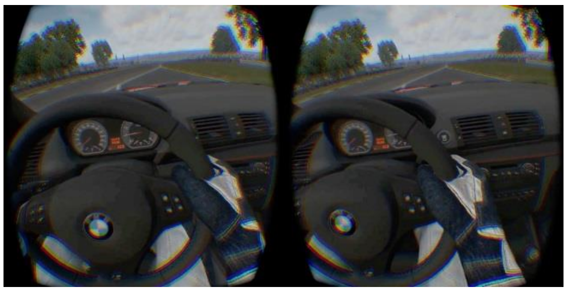


Figure 2.7: Project CARS through the Oculus Rift. (Rudderham, 2015)

Using different inputs and outputs together can have a variety of responses. Guo & Quarles (2012) found that using a motion controller with a VR headset in a racing game motivated the player to try harder when performing exercise, as the game was excerise based. However, in an interview with Makuch (2014), Shuhei Yoshida of Playstation Worldwide Stuidos stated that the game *Driveclub* with the *Project Morphus* headset was “kind of difficult and sickening," specifically at fast speeds.

2.5.2 – Procedural Track generation

One of the main reasons racing games are popular among developers and players is the rich amount of content these games require. One content inclusion that most players will look for in a racing game is the track list. This can be seen in some marketing tactics to highlight this, such as seen by the developers of Project CARS. Slightly Mad Studios (2015) have a separate page on their website promoting the game, highlighting the total number of different tracks in their game contains, to which they claim amounts to “over 35 unique locations and over 100 different courses!” This type of content is one of the most important visual aspects of a racing game, and can be the most time consuming task. This is specifically down to the process that each track must go through during creation, from the layout, to height adjustment and scenery addition, ensuring authenticity (if based on a real location.)

Loiacono (2012) points out that tracks can be generated to speed up development, mentioning three different approaches; theory-driven, data-driven and interactive approaches. Loiacono, et al (2011) notes that games that aren’t restricted to a specfic set of tracks are able to produce more content with the use of such approaches.

The game *Gran Turismo 6* offers the player the ability to create their own race tracks. This process is done through the *GT6 Track Path Editor*, offering the player to add features such as kerbs and scenery. This app was released in September 2015 by PlayStation Mobile Inc.(2015), and is available on andriod and iOS tablets. After creating a track, the user can then export it to the server, where they can download and race on it through their PlayStation 3’s. Figure 2.8A shows a track being created through the app interface, with figure 2.8B showing *Gran Turismo 6* running with the created tarck.

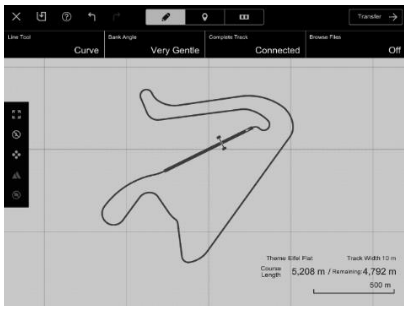


Figure 2.8A: Screenshot of "GT6 Track Path Editor" app. (Left) (PlayStation Mobile Inc., 2015).

Figure 2.8B: Screenshot of Gran Turismo 6 running the custom track. (Right) (Polyphony Digital, 2013)

2.5.3 – Artificial Intelligence

An important field of research for racing games is AI, as this gives the player an opponent to race with in this genre. When looking into AI, there are two distinct types that are generally talked about; behaviour and path finding.

2.5.3.1 – Behaviour AI

Certain actions that AI are expected to perform while racing is at the fore front of AI research in this genre. This is due to the complex challenge that tackling a problem such as this offers. Tang, et al (2009) investigated the use of behaviour based AI, while also implementing a neural network system. Results showed that the behaviour based system was more beneficial to use for more graphical demanding games as "efficient usage of computation resources is important for ensuring an uninterrupted game presentation" (Tang, et al., 2009) A suprising result of this research was that the behaviour based system was actually outperforming the neural network, which would be learning more and more about ideal situations to perfrom certain behaviours.

Research into fuzzy logic in order to control these behaviours was conducted by Fujii, et al. (2008). The authors aimed to use a waypoint system to get the car moving, with the fuzzy logic system deciding whether the AI should aim for the next waypoint in the system, or to aim for the waypoint that comes after that. Fujii, et al. (2008) stated that "appropriate waypoint selection is necessary for successful results in the car racing game."

2.5.3.2 – Path finding AI

Path finding in racing games is very important, especially for those aiming to give a realistic experience to the players. Multiple researchers have attempted to tackle path finding around tracks without the use of a way point system, the norm for developers, as Wang & Lin (2010) points out that "a simple waypoint AI system usually is implemented in car racing game."

One method of path finding proposed by Wang & Lin (2010) uses a new algorithm in which was designed for both 2D and 3D racing games. This method uses two collision points, one on the left and one on the right, and checks for the edge of the track or an object. The authors note that if one of these points is activated, it will steer in the opposite direction, while creating two new points, further apart than the originals, if both are activated. As claimed by Wang & Lin (2010), this means that "the right judgment to control the car to turn right or left."

Another method that was tried and tested was by using a set of sensors to navigate a race track in which the AI has no knowledge of previously. This system was developed by Butz, et al (2011) with the intention of entering this AI system into a competetion. A more in-depth look at this system is in section 2.6.2.

2.6 - Identified Racing Line Methods

Before development of the hybrid system can begin, current methods used by developers to create a racing line for AI to follow needed to be explored. This section highlights the methods which have been identified.

2.6.1 - Sector Based

The first identified method uses sectors and interfaces to define the racing line. This method, as documented by Biasillo (2002), is used to represent the racetrack for the AI by dividing them into sectors. Each sector starts and ends with an interface which "define the left and right most boundaries of the road and the possible driving lines" (Biasillo, 2002). This is effectively acting similarly to how a navigation mesh would serve the AI controlling non-vehicular enemies, feeding world data to that specific agent.

Each sector is capable of telling the agent important information regarding how to navigate the sector they are currently occupying (e.g. target speed, need for braking, turning speed, etc.) The racing lines are represented as a single point on each interface, which the agent aims towards. With these points linked together, as suggested by Biasillo (2002), the resulting line will be a racing line that the agents can follow. Figure 2.9 features a diagram of the sector based method, with a set of three interfaces creating a section of track, with the inclusion of two lines.

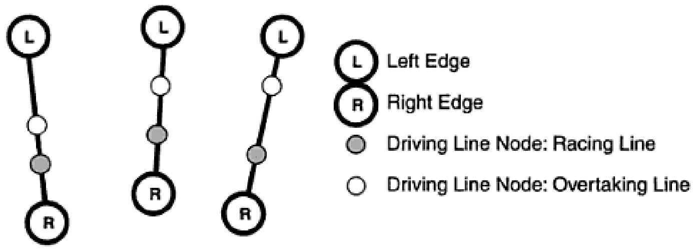

L
R
Left Edge
Right Edge
Driving Line Node: Racing Line
Driving Line Node: Overtaking Line

Figure 2.9: Diagram showing the Sector based system. (Biasillo, 2002)

Biasillo (2002) also mentions that the AI should be given information on specific points on the track, helping it make the best decisions, which can be collected from the sectors. Giving the AI information through this method means the developer can restrict how much data the AI should receive and how early the acquisition can be. This is important for AI in video games as they tend to be perceived by the public as having full knowledge of the world state, which leads to accusations of the AI cheating.

The information in which the AI can receive from the track sectors can vary depending on the type of game the developer is creating. One example given by

Biasillo (2002) is Brake/Throttle inputs. These inputs would be used in all racing games. However, some information, such as whether the sector contains a power up is usually reserved for games of an arcade nature.

2.6.2 - Sensor Based

The second method identified finds the racing line by using sensors. The AI in this method finds a suitable line itself, rather than the developers providing a racing line. On paper, this method has a lot of positives, from being able to dynamically adjust its line based on the environment, to developers being able to cut development time.

Butz, et al (2011) built a system that controls an AI car which can navigate around a race track, while avoiding other vehicles. They implemented a sensor based method to guide the car around, overcoming a stipulation in the *Simulated Car Racing Championship* (SCRC.) The SCRC was conducted on the *Open Racing Car Simulator* (TORCS), in which Butz, et al (2011) describes as “an open-source car racing environment with a rather realistic track and car simulation engine.”

Butz, et al (2011) used a sensor based method for getting the best racing line. Their system, which they named *COBOSTAR*, Butz, et al used nineteen separate sensors in which check for the track edge. Figure 2.10 demonstrates how the system uses the sensors to determine which direction to turn.

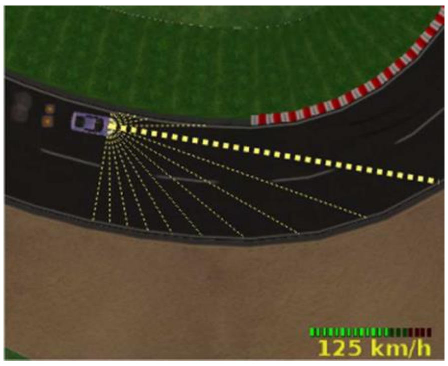


Figure 2.10: Diagram showing the sensor based method. (Butz, et al., 2011)

With these nineteen sensors, the AI is able to check for nineteen different routes which the AI can take. The sensors cover a 180° field of vision, which can aid the AI in the sharpest of corners. Butz, et al (2011) would use the information gathered the sensors to determine which line should be taken. The main direction that the AI should follow would be the sensor with the largest distance (the middle sensor in the case of Figure 2.10.)

However, there is one more piece of logic applied to this system. Before the AI steers, it will get the values of the two neighbouring sensors. From this, the three values are put into a formula, calculating the overall direction for the car to be aimed towards. This formula is crucial for the AI to know a target angle to steer the car towards, using values from the longest distance sensor (both the angle it is pointing at and the distance the sensor detects) and the two neighbouring sensors (distances only). The resulting value gives a target angle, which "slightly points towards the direction of the larger neighbouring distance" (Butz, et al., 2011).

The final part of this formula determines which direction the AI will slightly aim towards, giving a varying result every time. As it uses the left value for the comparison, a value for this section that produces a value less than 0.5 will result in the AI aiming slightly to the left. This target angle is used by the AI to steer the car towards the line it believes it should take, which is evolving to the situation in which the AI finds itself in.

2.7 – Overview of Artificial Intelligence in Racing Games

Looking back on AI in racing games, it's possible to notice how far this area has developed. Whether it's down to the type of setting the races take place in, or to technological advances that developers have taken advantage off to ensure their AI is top of the range.

2.7.1 – Historical Development

Motor racing games started making an appearance as early as the 1970's. One of the earliest examples is the game *Pole Position*. This game was designed to be placed inside arcades, which can be seen in Figure 2.11. *Pole Position* was released in 1982 and offered a single player racing experience. (Museum of the Game, n.d.)

Figure 2.11: A cockpit cabinet for Pole Position. (Museum of the Game, n.d.)

In *Pole Position* you drive a racing car around a track, dodging opponents, with points being scored the longer you survive and the more cars dodged. The opponents would move along one of two lines, either on the left or right side, as can be seen in Figure 2.12. The issue the developers would have developing this game was to ensure that the player had enough room to fit through any two opponents. Despite the two racing lines the opposing cars travel along, neither of the lines represented a line drivers would use.

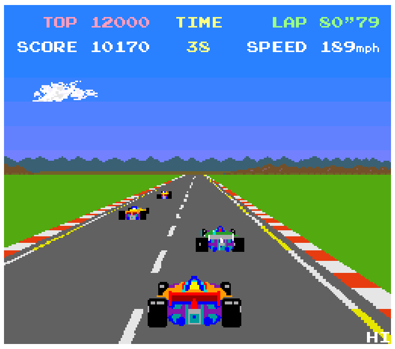


Figure 2.12: Screenshot of the game Pole Position. (Museum of the Game, n.d.)

FORMULA ONE GRAND PRIX was developed by Microprose and released in 1992. This racing game is considered to be "one of the first serious racing games" (Scullion, 2016), focusing on car setups and handling. This game was also one of the first racing games to use 3D graphics, by "making use of polygonal graphics long before it became the norm" (Scullion, 2016), allowing the individual locations of each car in the game world to be stored separately. As a result, the developers were able to accurately control the paths their AI cars could travel, as opposed to using two lanes. As a result, this game became one of the first to use a racing line, similar to those used by real life drivers. Figure 2.13 shows the AI in *FORMULA ONE GRAND PRIX* following the same racing line around a bend.

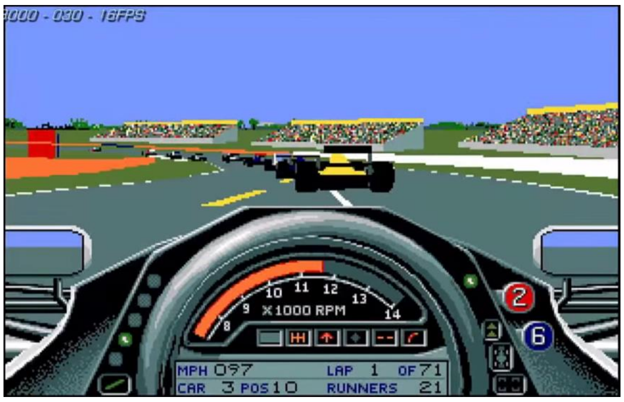


Figure 2.13: Screenshot of FORMULA ONE GRAND PRIX. (LemonTubeAmiga, 2012)

This game allowed for more realistic racing, as the player could chase an AI opponent at similar speeds. The game also kept track of the number of laps completed, along with the current running order. In a sense, this game was one of the first examples of an AI system in which was able to control the racing conditions.

A method that this game possibly implements is the sector system. This system defines the race track as a group of sectors, which are divided by interfaces where information for the AI to help it traverse the track is stored. This method didn't stop researchers from developing a more beneficial system for navigating an AI driver around a race track. One way in which they are testing these potential methods is by entering the SCRC which runs through *TORCS*.

The SCRC is a championship for developers to race their AI drivers against those of other developers. The championship sets the task for developers to get their AI to successfully navigate a track with the given "properties [which] are only partially accessible." (Butz, et al., 2011)

A sensor based method has come from this championship, which scans the track in front to find the longest line available. The AI does this by sending out "nineteen [sensors], which provide the information about track edges." (Butz, et al., 2011) With the gathered data, the AI finds the sensor with the greatest distance, and the neighbouring sensor with the greatest distance also, using a line in which is between the two sensors, slightly favouring the direction of the ray which the greatest distance.

This method would be beneficial to include into their games, as the system doesn't require as much set-up for each track, as compared to previous methods. This system would be beneficial for developers who plan on adding a great variety of tracks or to implement custom tracks, which can be generated by the player. *Gran Turismo 6* includes the option for players to fully design their own tracks on a tablet device, which can be seen in section 2.5.2.

2.7.2 – Track racing vs. Open World

One clear distinction that can be made about a racing game is how it offers its races. Racing games often use the traditional set up of selecting a course or track, and racing based on those fixed parameters. Recent racing games such as *F1 2014, Project CARS* and *Forza Motorsport 5* follow this racing discipline. Alongside this, games such as *Forza Horizon 2, Driver: San Francisco* and *Burnout Paradise* offer complete freedom in their races. Developers would create an open world environment to achieve this, offering a variety of routes to get from start to finish.

As these two types of games offer different ways to race, a few factors need to be considered when designing AI opponents. One of the main differences is how the AI navigates. For games with set courses, they can rely on using one of the systems identified in section 2.5. However, for open world games, they need to develop a system that can find the best route for any race, using any combination of roads. For example, *Burnout Paradise* offers an open world experience, with races typically offering more than one route to the finishing line, with the addition of various short-cuts.

Before the release of *Burnout Paradise*, the *Burnout* games have always been created along with pre-set tracks in a city setting. Making the jump from *Burnout*

Revenge to *Burnout Paradise* meant the developer *Criterion Games* had to rework the way the AI found its way to the finishing line.

Open world games can make effective use the A* algorithm, allowing the AI to find the shortest route to the finish line. This method was used by Wang & Lin (2012) on a set of traditional tracks, finding the shortest route around the tracks. This method, however, would be better suited for open world games as the tracks used by Wang & Lin (2012) don't offer variation in routes in one race as compared to open world racing games. During a race, the AI may come across a short-cut. From here, more logic can be applied, checking the possibility of using the shortcut. This could be done by using the A* algorithm, and seeing if it is possible to re-join the previously calculated route from the end of the short-cut. If possible, then the AI can proceed to use the short cut.

2.7.3 – AI and Online Multiplayer

An AI system was the only way in which players were able to race against a full grid for some time. Even with local multiplayer, players would still need AI opponents if they wanted to achieve a full grid of cars. A recent trend within the gaming industry is online multiplayer. This new mode allows players to face others, offering a new experience, as well as eliminating the need for AI opposition. Racing games also include this mode, allowing players to race other players, offering a lot of unpredictability in every race.

Despite the eagerness to play against others by many players, there are some that still prefer racing against some form of AI. This is evident from developers that still focus on a single player component, allowing for the players to race against AI offline. Some games like *F1 2015* by Codemasters (2015) encourage players to "immerse yourself in the new Championship season, [to] test yourself to the limits in Pro Season" while racing against the AI. *Project CARS* also accommodates these wishes, providing the player with both a career mode and the ability to set up races where they can race against the AI (Slightly Mad Studios, 2015).

Few racing games allow the host of a multiplayer session to include AI opponents. This feature is mainly utilized to fill any free spaces left, filling the grid. Most games do offer a single player component, which does require AI opponents. One notable series in which does this is the F1 games developed by Codemasters. A main feature in these games is the career mode, which pits the player against other drivers that are controlled by AI. Codemasters have also added another mode, allowing players to team up with friends online and play through the championship against AI drivers.

2.7.4 – AI learning from the Player

To produce more believable drivers, developers can design an AI system to observe the behaviour of the player, building a behaviour profile through these observations. This technology is available in *Forza Motorsport 5*, which tasks the player with a race when starting the game for the first time. This race is used to record data, allowing the game to determine a difficulty level. This difficulty level will then be fine-tuned with the next series of races, each one still learning the play style of the player. The AI opponents are representations of other players, exhibiting characteristics such as variation of braking and the top speed reached.

According to Microsoft (2015), the five main characteristics which the AI, known as '*Drivatars'*, will learn from the player are: consistency, line, entry speed/braking points, apex speeds/positions and exit speeds/acceleration points. This method can generate multiple behaviour sets for the AI to follow based on the data gathered; giving players a more varied racing experience.

Players can continually train their *Drivatar* by completing lessons. Each lesson involves the player doing three observed laps, while the AI records data about the various techniques used. These lessons can be repeated to improve the AI, meaning that the player can evolve their *Drivatar* to become a better racer. Further training can be applied through free training, allowing the player to use any combination of car and track.

Chapter 3 – Planning the Hybrid System

3.1 – Introduction

Before developing a prototype of a hybrid system, the artefact needs to be planned, defining what needs to be implemented and what is more important rather than what isn't necessary. It is crucial to get working implementations of the two systems that will be used in the hybrid system, contributing towards the system planning; highlighting what features can be used.

3.2 – Implementing Current Racing line methods

As part of the research into AI navigation in racing games, two methods were implemented to get a better understanding of how the systems worked. The methods for this section were the sector and sensor based methods.

3.2.1 - Sector Based Implementation

To speed up the process of setting up the track, a prefab, which "acts as a template from which you can create new object instances in the scene" (Unity, 2015), was created within the Unity game engine, storing the interface that divides the track into sectors. This allows any new interfaces to be added quickly to the scene. However, a flaw in this setting up process was realised. Since Unity doesn't track the order each interface was created, this would result to having to manually assigning a number to each interface. This ensured that the waypoints connected in the correct order during runtime. The setting up process of the track took approximately eighteen hours.

The next task was getting the AI to move to and follow the waypoints. The AI was programmed to travel at a constant speed, with the objective of the AI successfully travelling between the waypoints in the correct order. The AI was then modified so that it rotates toward the target in a gradual manner, making its turning behaviour more authentic.

The system keeps track of the waypoint it should aim for with use of two variables; one to store the next interface for AI to hit, and to determine which line they are using. One of the first tasks the AI completes is finding every interface, so it knows when it is approaching the last waypoint in the sequence, allowing it to aim for the first interface. Figure 3.1 shows the AI car aiming towards the racing line waypoint within the next interface in the sequence.

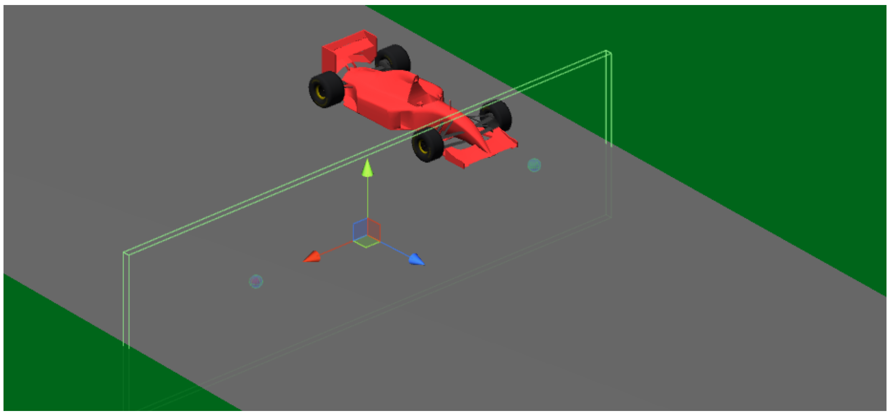


Figure 3.1: AI car aiming towards the racing node of the interface.

With the system following the interfaces in the correct order by targeting the correct waypoints, work on more realistic movement would begin. This part of the AI system will be affected by physics, whether it be as a result of natural behaviour or as player intervention (e.g. a crash). Unity provides its own physics engine, which was used to achieve this feature.

The AI uses dedicated values to determine whether it should accelerate, brake, how much steering it should apply and the speed it should target. To make the system easier to transfer to multiple tracks and projects, target speeds are embedded into the interfaces, allowing the AI can gather the values and compare them to its own. For the AI to “have a better understanding of the environment, relevant data should be stored within each sector.” (Biasillo, 2002)

With data being fed to the AI from the interface it is targeting, it will compare it alongside its own data, making a decision based on these results. For example, if the AI finds *currentSpeed* to be lower than *targetSpeed* from the interface, then the AI knows that it needs to be accelerating. This highlights another issue with set-up method of the track. Although interfaces can be updated to add the variables via editing the prefab, it meant having to manually assign a target speed to every waypoint.

One issue was discovered with this method, resulting in inconsistent paths being taken by the AI. This issue can be related to the steering motion, as the gradual steering is set to a locked value at each frame, however not instant. The AI would be able to steer the car more than required, meaning some strange behaviours with the AI repeatedly steering left then right. Examples of these varied lines can be viewed in appendix A, and in figure 3.2, particularly on the second to last straight.

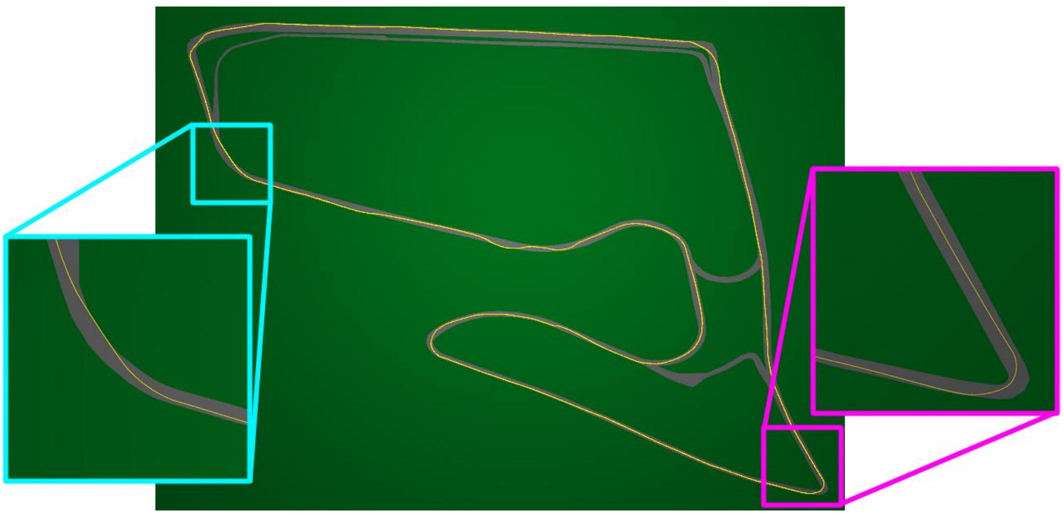


Figure 3.2: Line Drawn by the AI of the Sector based system. Notice how the line taken by the AI in the corner highlighted in blue isn't a consistent curve, with the AI going straight for the corner itself.

3.2.2 - Sensor Based Implementation

The sensor based system was a method used by Butz, et al (2011), and was capable of winning races at competitions. As this path finding method hasn't been discussed by many researchers, this experiment will be the main source of understanding how this system navigates a track.

To prepare the AI to navigate the track, the car was fitted with a set of sensors. The sensors were placed between the front wheels of the vehicle, and spread out like a fan, with equal angles between each sensor. Unlike the system implemented by Butz, et al (2011), this version isn't spread out at a 180° range. This is due to human perception of sight, a topic in which Conroy & Wyeth (2011) explains in detail. Despite the suggestion of a 120° range, a range of 170° was used. By using this range, the AI have a greater visual range to look for a line. This greater range is designed to mimic the head and eye movement of the drivers when they look towards the apex when cornering, while also restricting the movement to stay in line with restrictions in visual range that the car may present. The head movement has been implemented for player cameras while locked in cockpit view, giving the player a different view of the track ahead.

A study conducted by Land & Tatler (2001) observed the head and eye movements of a racing driver to determine the differences they take in observing the surrounding environment as compared to a normal road driver. This research showed that in the most extreme cases, the highest angle a racing driver would turn their head at was 20 degrees, while the mean rotation of the head for that particular corner was around 8 degrees. The eye movements for the same corner would have a range between five and fourteen degrees, resulting in 2.5 to seven degrees of extra visual range. By taking the twenty degree head movement into account of it being the most extreme case, the use of the second highest degree of rotation, eighteen, would be used. With the highest eye movement from this corner (seven), we are able to get an extra twenty-five degrees either side for the driver to visually see. Figure 3.3 shows this behaviour being exhibited in a real life scenario, as the two drivers turn their heads toward the corner they are navigating.

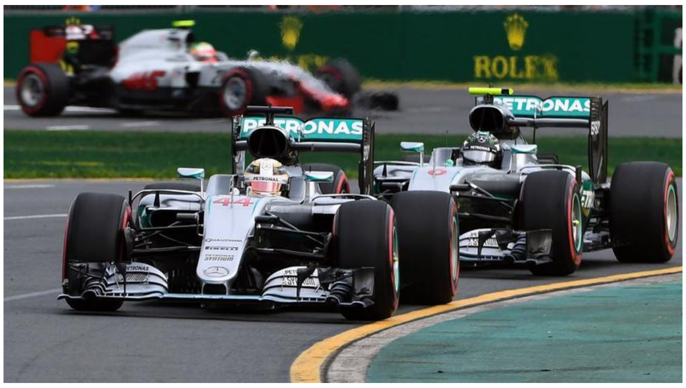


Figure 3.3: Drivers exhibiting the head movement that the view range is simulating. (Most Reliable Car Brands, 2016)

After implementing the sensors, they needed to be able to detect the edge of the race track. After consulting the research by Butz, et al (2011) to understand how this was accomplished, it was the research by Wang & Lin (2010) that provided a solution.

Wang & Lin (2010) implemented a system that acts similar to the sensor system used by Butz, et al (2011), despite the sensors being points at the front of the vehicle. These points check for collisions with another game object, adjusting the cars direction in response to any detection. One of the game objects searched for defined the track edge, represented as an object of colour (as the system created was colour based). Fortunately, Unity's rendering system allows meshes to be hidden from the user, while still maintaining its physics body. The physics component can also be adjusted so that only sensors can have any interaction with it. Figure 3.4 shows the sensors reacting with this game object, which cannot be seen. The longest ray is represented by the yellow line, while the second longest is represented by the blue line.

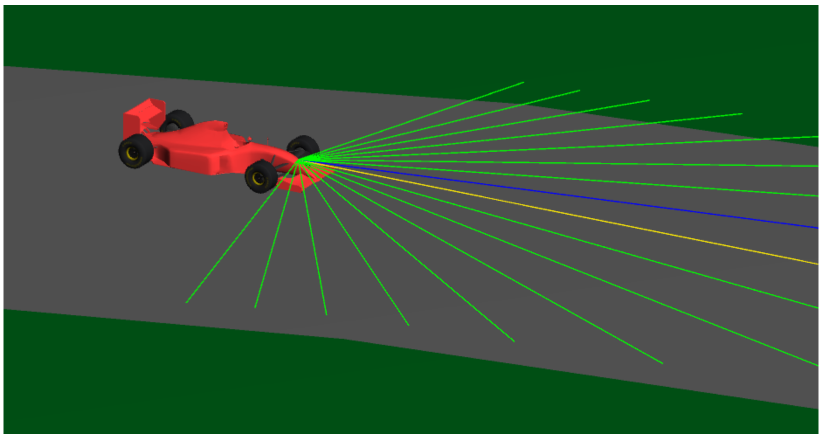


Figure 3.4: A visual representation of the sensors system. The lines being drawn from the car represent each of the 19 sensors, with the blue line representing the longest distance while yellow represents the second longest.

The AI was assigned a constant speed to test if it can travel around the course successfully. The AI was to rotate to match the rotation of the sensor that was determined to have the longest distance to the track edge. The first issue with this system was realised at this point, with a solution found after reading further into the paper by Butz, et al (2011). When cornering, the AI would end up with part of the vehicle off the track. This was where the sensors would pick up another part of the track, rather than the corner it was at. As a result, the AI would face that section, cutting out part of the track.

Butz, et al (2011) mentioned that the AI targeted a position that is the result of the longest detected sensor distance and the longest neighbouring sensor. Testing this system without the AI controlling the car revealed that the two longest sensors would be next to each other in the majority of cases. The AI was coded to use a vector of these two sensors, with the sensor with the longest distance slightly favoured. However, a precaution was taken due to further testing showing that the issue still occurred. The faces of the object from the research by Wang & Lin (2010) were reversed and added to the system. The intention of this was to set the sensor value to zero if it had registered a hit with this new object.

After the modifications proved successful, the final step was for the AI to accelerate and brake appropriately. The AI in this system had to determine if it was appropriate to apply the throttle or brakes, via the collected information of the sensors. This was done by calculating the distance to the target it uses to aim the car. The acceleration and braking was all controlled using the physics engine, just like the implementation of the previous method. Figure 3.5 shows the line the sensor system was able to draw.

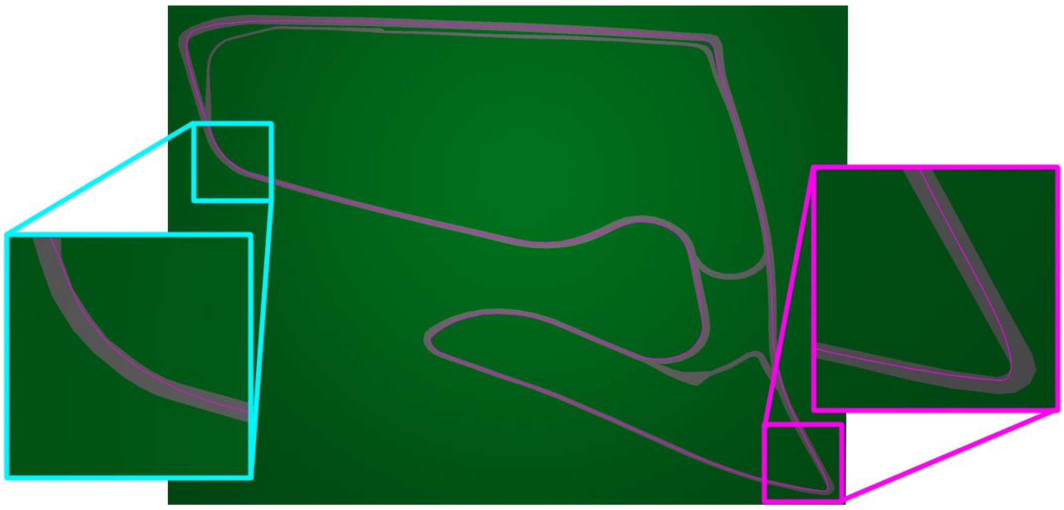


Figure 3.5: Line Drawn by the AI of the Sensor based system. Notice how the line taken by the AI in the corner highlighted in blue is a consistent curve than that of the sector method, while the corner highlighted in the pink box shows the car entering and exiting the corner within the centre of the track rather than the edge.

To further test this system, the sensor system was set up for a second, smaller track. Figure 3.6 shows the sensor AI navigating a smaller track, with successful results.

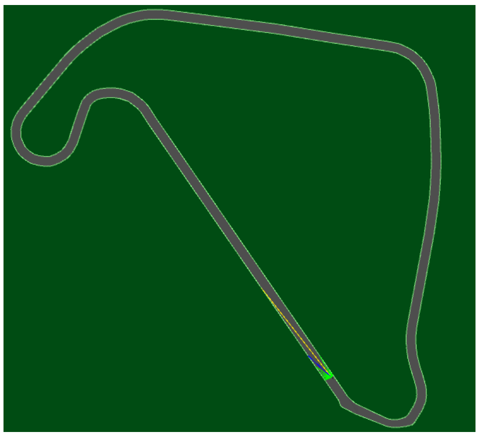


Figure 3.6: Sensor based AI navigating a second track which was set up to determine if the method was able to navigate multiple tracks.

This sensor based method was originally tested within TORCS, with documentation of the system didn’t explicitly state certain features of how the AI functioned. One of the main problems that had to be tackled when setting up this method was discovering the track edge. With the aid of the research conducted by Wang & Lin (2010), this implementation of the sensor based method is the first known version outside of the source paper, with another advantage of being developed outside of TORCS and into a commonly used game engine (Unity.)

3.2.3 - Comparing the Racing Lines

With the racing lines of the two systems drawn and stored in separate text files, they can be loaded and analysed in a separate program. Figure 3.7 shows the lines drawn in the previous two sections loaded onto the track, with the sector system being represented with the yellow line, while the sensor method is drawn with the pink line.

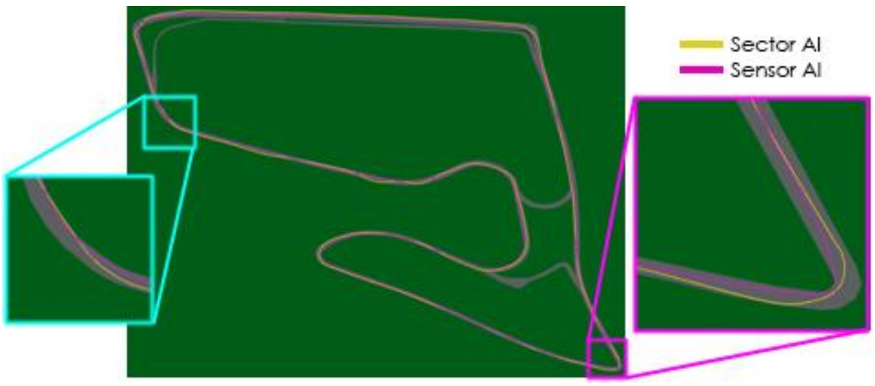


Figure 3.7: Lines drawn by the sector and sensor methods. The two highlighted corners methods show the strengths and weaknesses of each of the two methods. The sector method is able to use appropriate lines through the straights as well as corner entry and exits (yellow) while the sensor method is able to take the cornering lines better, finding the apexes every time(pink.)

From figure 3.7, it can be seen that the two systems take different lines. The sensor based system loaded faster due to a lower node count, which suggests that it had a quicker lap time than its sector based counterpart. Also, as the sector based system requires being set up manually, the line created is based on the line in which drivers usually take around this course.

The first noticeable difference between the two lines is on the start/finish straight. The sector based AI takes a line in which it sticks to the outside of the track, while the sensor based AI took a line that was closer to the inside. This affects the path for the AI for the duration of the straight, leading up to the first corner. While the sector based line sticks to the outer, the sensor line begins to move towards this edge, but doesn’t drift far from the centre of the track. Although the behaviour displayed by the sensor AI isn’t what would be expected in a real life scenario, it does find a line rather than just sticking to the inside of the track.

Both systems show good lines when cornering. The sensor AI aims for the apex of the corner, despite the sharper angle caused by its entry line. Comparing this to the sector line, the AI also aims for the apex. However, the corner is less sharp, due to a better line into the corner entry. These behaviours can be seen in figure 3.8.

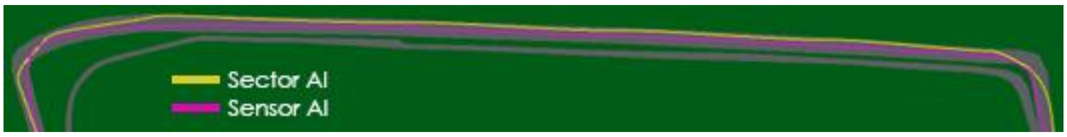


Figure 3.8: The lines of the two systems on the main straight. Notice how the line taken by the sector based method (yellow) is positions towards the outside of the track, readying the AI to take the next corner, while the sensor method (pink) is positioned more towards the middle of the track.

The lines exiting the first corner behave differently. The sector based method takes a wider exit, a characteristic seen by the same method when entering the corner. The wider line allows the car to complete the corner at a greater speed compared to the sensor AI. The sensor AI takes a sharper line than the sector AI, and this is due to the AI having a bad entry. Despite this, it managed to turn more, being closer to the inside edge of the track than the sector based method. Although it’s not surprising that the sector method is following the racing line, it was unexpected for the sensor based system stay away from that line. However, the sensor AI found the apex of the corner, despite its poor entry line.

This was a trend that repeated itself for the remaining straights and corners. Entry into a corner was better for the sector method, while the sensor AI was able to make the apex, regardless of the poor entries. The next corner, turn two (see figure 3.9) is the sharpest of the track, with the AI behaving the same as it did at the first turn.

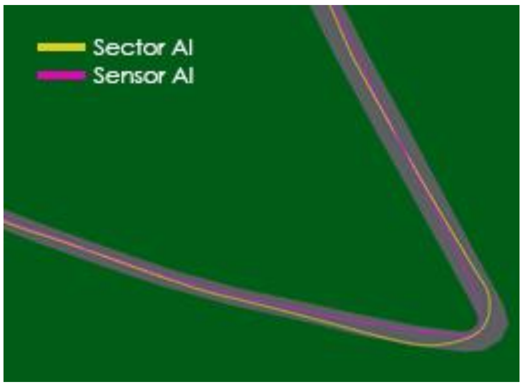


Figure 3.9: The two lines drawn at turn two. The sector based method (yellow) has a wider entry and exit, while the sensor method (pink) takes a tighter line, but is still able to hit the apex.

A behaviour that was displayed by the sector based AI is a slight variation in the path it takes. This behaviour usually happens on a straight, with the AI car randomly turning left or right. This behaviour can be explained as the AI tries to adjust for the next waypoint, but turns too much, meaning it would need to correct itself. This would lead to the AI then turning in the opposite direction, running the risk of applying too much steering again. Figure 3.10 shows an example of this, where the sector based AI turns left, off the track before correcting itself, before having to turn left again to correct itself from a second over steering moment.

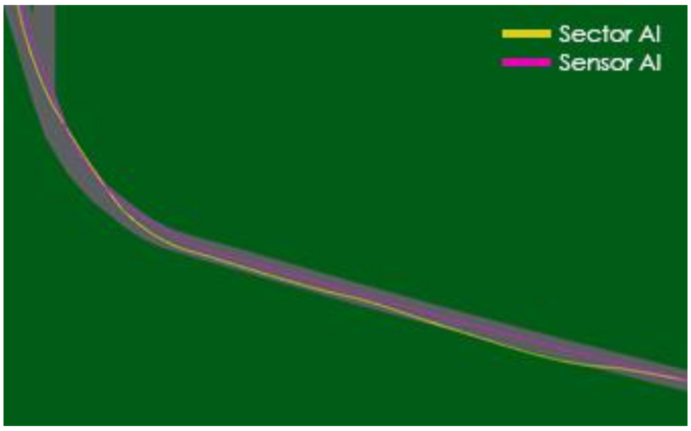


Figure 3.10: Slight variation in the line drawn by the sector based system (yellow), where it can be seen that it is not straight, with the line moving left and right, with the AI going off track for a short time.

3.3 - User Racing Lines

To achieve a more realistic racing line for the AI to follow, the racing lines of a number of users were gathered to understand how believable the line the AI drove was compared to a human. An application was built to allow the users to drive around the track, with the same physics as the AI, storing their fastest lap time, position, speed and inputs at each frame into a text file.

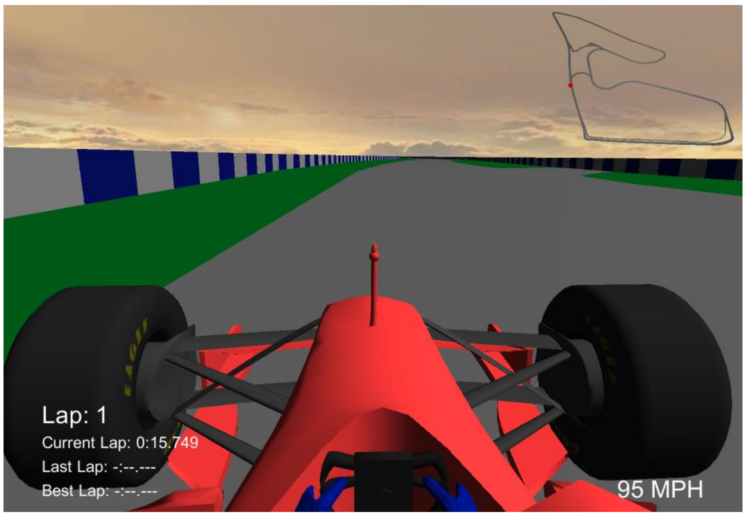


Figure 3.11: A screenshot of the demo that captures racing lines.

When saving the line to a file, it wasn't necessary to know how many laps the user completed. The application would store a maximum of two laps worth of nodes at once. These nodes are stored within separate game objects called "*currentLap*" and "*fastestLap*." The "*currentLap*" game object is created at the start of every lap, with a node created and moved inside every frame. If the lap becomes the fastest time, it is renamed "*fastestLap*" while any other game object with the same name is destroyed. If the lap isn't the fastest, then the game object is destroyed.

Once a lap has been completed, the option to save the lap to a file becomes available. Unlike the AI demos, the save destination can be changed during runtime, and with multiple saves also being possible.

3.4 - Reading the Racing Lines

After collecting the racing lines from human players in the demo, they could be used for comparison against the two AI methods implemented beforehand. We can identify a number of points, such as the variations of lines taken and which system closely resembles a player line, as well as the accuracy of each line.

For this thesis, the racing lines of three groups of people, each with different levels of experience; experienced with racing games (Experienced(RG)), experienced with games but not with racing games (Experienced (NRG)), and not experienced with games (Non-experienced). Having three volunteer groups with different levels of experience enables the evaluation of lines taken by players with varied levels of skill, fitting into the qualitative methodology. The results should show that the Experienced (RG) players will not only be faster around the track, but also take better lines.

3.4.1 – Experienced Players (Racing games)

The first group of players that will be compared alongside each other are those that are experienced with racing games. This comparison will look at the lines each of the five players took, and deciding on the best line to represent the group based on decision influenced by certain parameters.

Player	**Lap Time**
1	1 min 16.098 secs
2	1 min 13.943 secs
3	1 min 16.835 secs
4	1 min 14.471 secs
5	1 min 14.599 secs
Average	***1 min 15.189 secs***

Figure 3.12A: The lap times of the experienced players (RG).

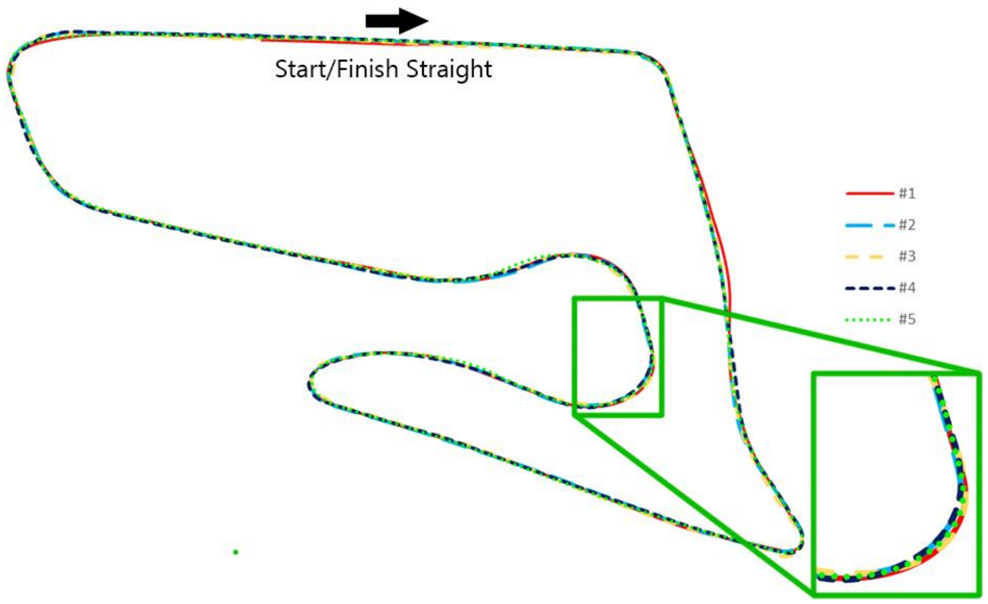


Figure 3.12B: Racing lines of the experienced players (RG).

From the results gathered, two interesting notes can be made; the lap times between the five players are close, and the lines taken by the player to achieve those times are very similar. These two traits were expected to be shown here, as a result of the players having the experience of figuring out the best line to be taken, as well as points to brake, and the speeds they can take each corner at. When deciding which line to use to represent the group, the process of elimination will be strict on lines that show signs of running wide or corner cutting, as well as laps that don't follow a similar line to a majority of the others on straights.

The first line which falls foul of this rule was produced by player one. This line is problematic in a couple of areas, on the start straight where the player started the lap from the middle of the track rather the track edge like the other players, as well as on the second straight, having to turn more sharply to get into position to navigate a small curve in the road ahead. This line also follows a line through the final corner which seems tighter than the other players. By omitting this line, the new average of this group comes to 1 min 14.962 seconds.

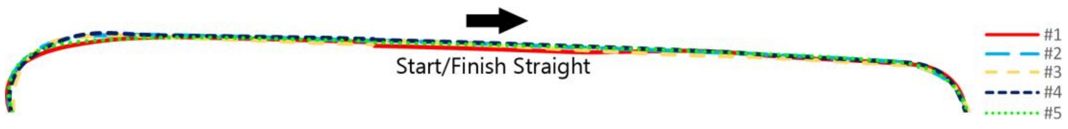


Figure 3.13A: The lines on the first straight by the experienced players (RG).

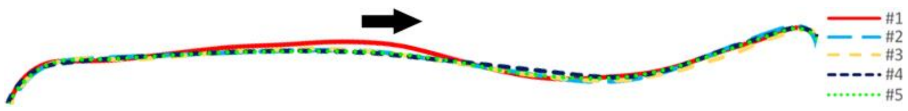


Figure 3.13B: The lines on the second straight by the experienced players (RG).

The next line that can be eliminated from contention from this is the third players. Just like the first player, this player also has a few issues that can knock itself out of contention. Firstly, the gap between itself and the average is 1.873 seconds, while the fastest time is 1.019 seconds ahead of it. This gap is almost double the size, with the gap between itself and the second slowest time being 0.363 seconds faster. From figure 3.13A above, it can also be seen that on two occasions during the straights, the player moves towards the right of the track, while others remain to the left. Several corners (including turn two) also show the player taking a tighter line, which causes them to get a bad exit line.

The last three lines do also show inconsistencies with each other, with each line matching at least another in most areas, while not doing so in other places. As this is a general pattern, a decision to stop the culling process with these three lines was made, and a new average time was calculated. The average for these three lines is 1 minute 14.337 seconds. In this situation, the closest lap time will be used to represent the group for the comparison. This decision making process will be applied across all of the groups. For this group, the line by player four is the closest to the average, resulting in this line representing the group.

3.4.2 – Experienced Players (Non racing games)

The second group looked at how experienced players with little to no experience with racing games fared. From here, we should expect players to be slower than the more experienced players, but still able to set a lap time which is not too slow, as they have the experience of using their chosen method of control.

Player	**Lap Time**
1	1 min 19.685 secs
2	1 min 16.737 secs
3	1 min 45.546 secs
4	1 min 26.402 secs
5	1 min 22.885 secs
Average	***1 min 26.251 secs***

Figure 3.14A: The lap times of the experienced players (NRG).

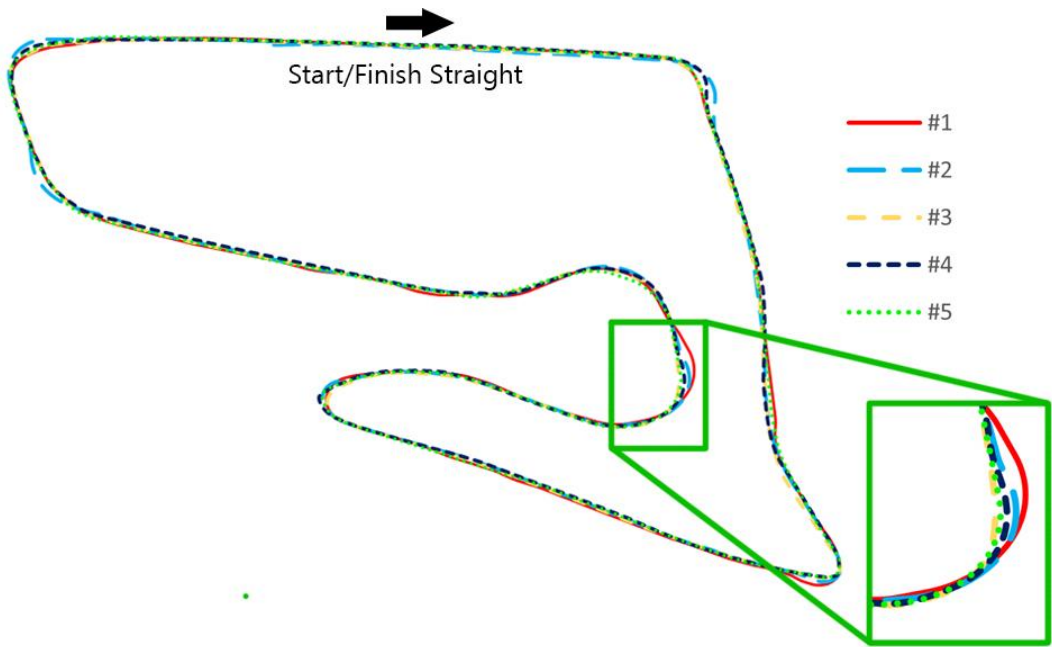


Figure 3.14B: Racing lines of the experienced players (NRG).

From the lap times shown, we can see that majority of the players are within ten seconds of each other, with the lap time of player three being around twenty seconds slower than player four, who recorded the second slowest time. This time difference had an effect on the average time, resulting in it being just slightly faster than the fourth player. It should be noted that despite the returned result stating the use of a keyboard, the player reported that they did use a controller to accelerate and steer, while having to use the keyboard to brake.

This result can be deemed inconsistent with the other data collected, which were completed with the use of one control scheme (whether it was with the use of a controller or a keyboard.) As a result of this inconsistency, this lap time can be removed. The average lap time of the group now becomes 1 minute 21.427 seconds, with the average is now nearer to the middle of the ten second range identified previously, providing a more representative average for the group.

The next step is to identify the line that will represent the group in the next comparison stage. As these players don't play racing games as often, it should be expected that the lines that they take won't be as *clean* as their more experienced counterparts. To allow this expectation to be realised, each line will be allowed to have no more than two instances where the player ran wide on the corner exit. This will be judged by looking at the lines of all the valid lines, and seeing how closely they are grouped, with any odd lines being judged to run wide. It should also be noted that track cutting will be treated as running wide also.

By looking at the line comparison, the first lap we can eliminate was produced by player two. On this lap, the player manage to run wide at three separate corners; turns 1, 7 and 8. As this goes over the limit of two wide lines on corner exit, the line can be omitted, and a new average can be taken. The line by the first player runs wide on the exit at turns 2 and 4, meaning it is within the excepted rule. Out of the remaining lines, only line 4 has an offending corner, with turn 1 being the location of this instance.

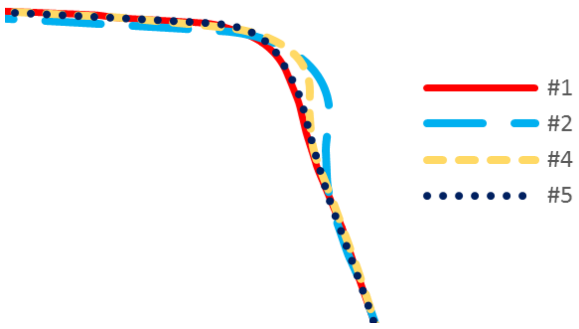


Figure 3.15: Racing lines of the experienced players (NRG) at turn one. For half of the samples used, this proved to be the trickiest turn in terms of corner exit, with lines 2 and 4 being the offenders in this case.

With the remaining three lines, a new average can be determined. What is interesting to note is that the two lines previously removed were the fastest and slowest lap times, meaning the group of players that will be focused on for the line decision will be the three players in the middle of the sample. The range between these three players is within 6.717 seconds, with the average time of 1 minute 22.990 seconds. This average closely matched the time set by the fifth player, whom set a lap time of 1 minute 22.885 seconds, meaning this line would be used to compare the lines taken.

3.4.3 – Inexperienced Players

The final group of players that were tested had no prior experience of playing computer games. These players were selected based on the amount of time they have spent playing computer games, which should result in slower lap times, as well as a variety in the lines taken between each player.

Player	Lap Time
1	1 min 58.620 secs
2	2 mins 11.310 secs
3	3 mins 11.569 secs
4	1 min 46.726 secs
5	6 mins 18.113 secs
Average	***3 mins 5.267 secs***

Figure 3.16A: The lap times of the inexperienced players.

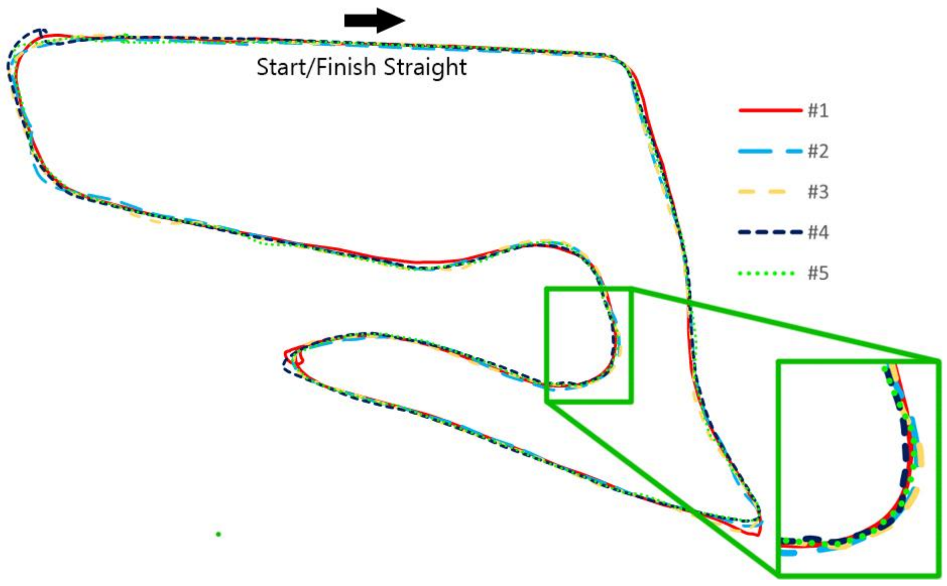


Figure 3.16B: Racing lines of the inexperienced players.

From the data presented above, we can make a judgement on the behaviours of inexperienced players. The average lap time of the five players in which have taken part is 3 minutes and 5.267 seconds. What is particularly strange about this average is how it closely matches to the second slowest lap time, being quicker by around 6 seconds. This can be attributed to the time set by the fifth player, whose lap time was almost twice as long as that of the player three.

Based on these observations, we can omit the lap time from the fifth player, producing a new average lap time of 2 minutes 17.056 seconds. Despite this average still be slower than the second slowest time (now set by player two), the gap in lap times between the two slowest cars are not as big as the previous comparison, meaning the average is more representative for the group.

Just like the previous two groups, one of these lines will be used to represent the whole group when it comes to comparing the lines of the overall groups. For this group, a few things need to be considered before selection. Firstly, leniency of the lines and accuracy of track navigation needs to be greater than that given to the previous two groups. This is due to the lack of experience these players would have with playing games. This leniency will be based on whether the player took corners correctly, or whether they managed to cleanly set a lap, allowing for one crash per lap (if any.) Figure 3.17 shows the second corner of the lap, where three players ended up overshooting the corner or ran wide on the exit, with the fourth player being the only person to navigate cleanly without overshooting the corner.

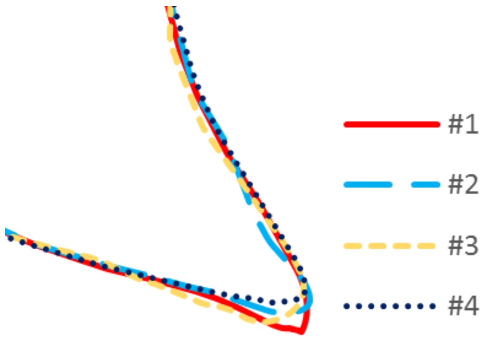


Figure 3.17: Racing lines of the inexperienced players at the second corner.

When looking at the racing lines, player three had the best when staying on track. The player was also consistently taking a line into the corners win which would be similarly matched by at least another player. However, the downfall of this line is the lap time. While being included in the new average time, the lap time still falls behind the second slowest lap by 60.259 seconds, while the fastest and second slowest is separated by 24.584 seconds. Based on this time difference, the average lap time could be recalculated, with the exception of player three's lap time. This reasoning could be further strengthened by looking at the input values for the player, where they would be below full acceleration for majority (if not all) of the recorded lap.

The average for this group now stands at 1 minute 58.885 seconds. However, two of the three remaining samples set a lap time with the player suffering a crash during the lap. However, it was these players in which ended up setting the faster times. The leniency for this group that set earlier took into account the mistakes that players of this group were prone to make. This decision was made based on the original sample size, where 3 out of 5 players ended up crashing. So based on this, no more exclusions from the data should be done.

The final decision on the line the choice was to take the average of the remaining sample group and compare it to the lap times. This makes the selection process as consistent as possible, ensuring that no bias in the decision making was present. In this case, the lap time from player one closely resembles the average time, with a difference of 0.265 seconds being present. As a result, the line taken by player one would represent the inexperienced player group.

3.4.4 – Comparing the groups

With lines chosen to represent each group, comparison of the three racing lines can be conducted. As discussed throughout the section, the chosen samples were selected on how representative they are to their group, with different restrictions across the groups to accommodate the different experience levels. Figure 3.18 shows the lap times that the selected players.

Player	Lap Time
Experienced (RG)	1 min 14.471 secs
Experienced (NRG)	1 min 22.885 secs
Non-experienced	1 min 58.620 secs

Figure 3.18: Table showing the lap times of the three players.

With the lap times of the three players set out into a comparative table above, it will be possible to compare them to the times from the AI. The following table (figure 3.19) shows the lap time of each system.

AI System	Lap Time
Sector System	1 min 12.042 secs
Sensor System	1 min 08.388 secs

Figure 3.19: Table showing the lap times of two AI systems.

The results show that the Experienced (RG) player was quickest from the selected samples, setting a lap time of 1 minute 14.471 seconds, 8.414 seconds faster than the Experienced (NRG) player, while being another 35.735 seconds faster than the

Non-experienced player, whom crashed on the lap they set. These results provide a better understanding at how different lines affect the time taken to complete the same distance.

For example, the Experienced (RG) player would position themselves on track where minimal steering would be required, slowing them to travel at faster speeds around corners, which would save them time. This is compared to the Experienced (NRG) player, whose positioning of some corners required more steering to be applied, resulting in slower corner speeds to get round without going off track, increasing the lap time. The differences in the lines can be seen in figure 3.20.

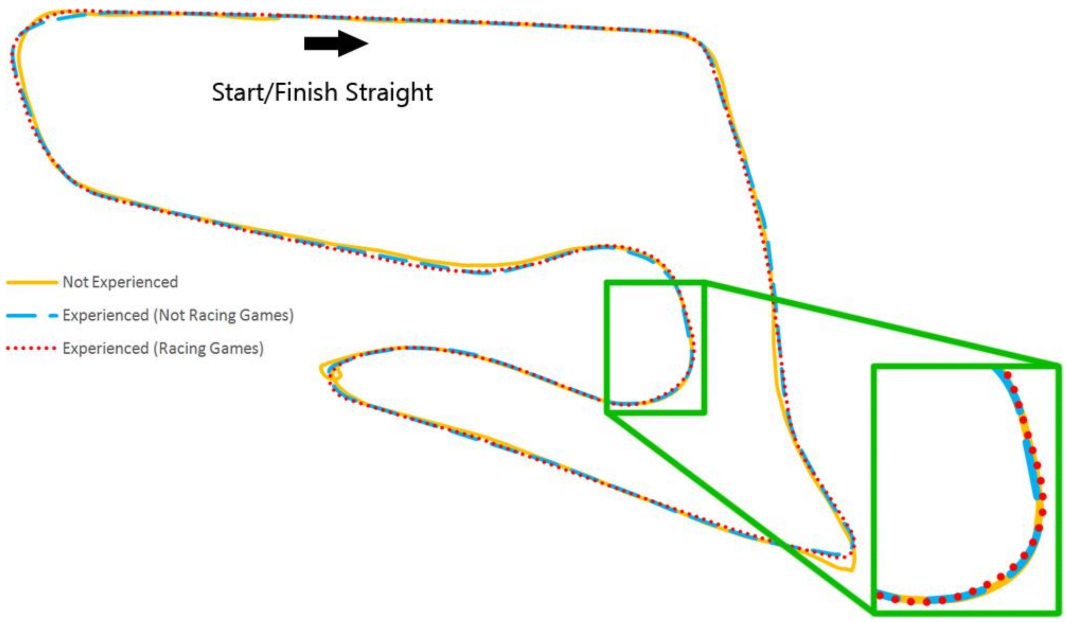


Figure 3.20: The lines taken by the three players. From this diagram, it can be seen that the line taken by the more experienced player (red, dotted) is more smoother around the track, with the line taken by the least experienced player (yellow, solid) having areas where the player would take an odd line, which is a result of the player crashing.

A few interesting details can be taken from this diagram, most noticeably the apexes taken by the three players. The second corner is a good example of showing the differences, where it can be seen that out of the Experienced (RG) player has better knowledge of the situation, managing to make the corner with minimal to no track extension. This result comes from a set of important factors; approaching speed, positioning and point of steering. Comparing this to the line taken by the Non-experienced player, the data shows that this player carried more speed leading up to the corner, resulting in them overshooting. The Experienced (NGR) player managed to hit the same apex as their more experienced counterpart, but had a tighter angle to turn due to their positioning at the corner entry. A closer look at the lines taken for this corner can be seen in figure 3.21.

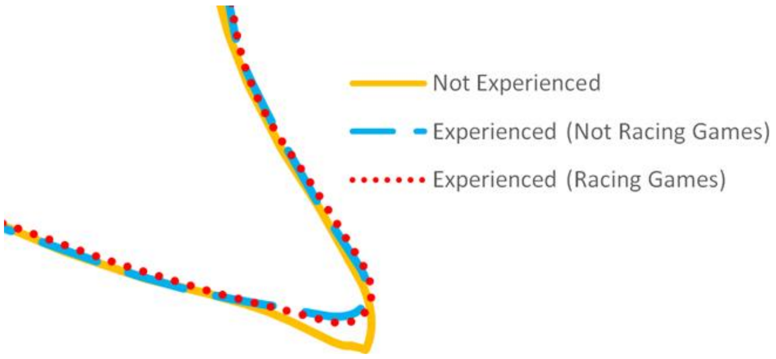


Figure 3.21: Lines taken by the three players at the second corner. From this corner, we can see the differences in how each player navigated the turn. The more experienced player (red, dotted) takes a smoother line, with less steering taking place on approach and exit, gradually reaching the track edge on exit, giving a straighter and quicker line, whereas the least experienced player (yellow, solid) overshot the corner.

Comparing these lines to those of the AI systems allows understanding of how close these systems are to matching a line which can be deemed believable to the player can now be completed.

From the five lines produced, the sensor based system managed to navigate around the track in the fastest time, setting a time that is just over six seconds quicker any player. The sector system also posted a quicker time than the players, with the difference being smaller than the sensor based system.

Based on lap times, the sensor based system is a lot quicker around the track. This can be explained by simply looking at how it works. Players were able to control their steering input with more precision using a thumb stick on a game pad. However, they had to contend the increased steering circle, which grew as they went faster. This becomes a problem for cornering as the tighter turns will require the player to go slower. The sector method also has to deal with this steering arc.

The steering in the sensor based method allows the AI to take corners at much greater speed. This is due to it not being restricted to a steering arc, but rather just the sensors. Every frame, the car would rotate from the position it is currently at, rotating itself to face the correct direction, as determined by the sensors. This means that it will be able to make any corner with relative ease, hitting the apex. However, a consequence of this is that with slower hardware, and despite the use of delta time ("time between each update or fixed update call" (Unity, 2013)) to ensure the system runs at the same values across all hardware, the AI can oversteer off the track. Appendix B shows an example of a line produced by the sensor system with this occurring where the AI takes wider lines out of select corners. Appendix B also shows that the AI on this hardware will not finish a lap when recording a line.

To look at the believability of the two AI systems, their lines will be compared to the line representing the most experienced group. To determine if the difficulty is competitive, the times of these three laps will also be the focus of this comparison.

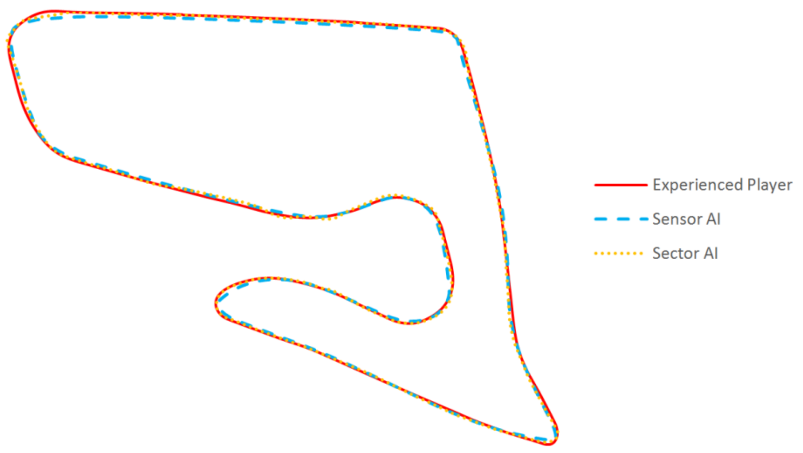


Figure 3.22: The lines taken by the player and the AI systems. From this figure, it can be seen that the Sector AI (yellow, dotted) is matching the experienced player (red, solid) for majority of the straights, while the sensor (blue, dashed) is matching the apexes taken by the player.

Figure 3.22 shows the lines of the player and the two AI systems. We can see a clear difference in the line taken by the player compared to the AI, with the sector system matching closest to the player's line. Reasons for concluding this is due to the lines taken through the corners and down the straights.

The sector system takes the straights and corners similarly to the player. This can be seen on a number of straights and corners, as the AI takes a wider position into the corners, just like the player. The sensor based AI follows a different line, where it sticks more to the middle of the track, with the exception of the corners. Figure 3.23 shows a closer look at a series of corners, showing how the sensor AI finds apexes while the interface and player were slightly off the mark.

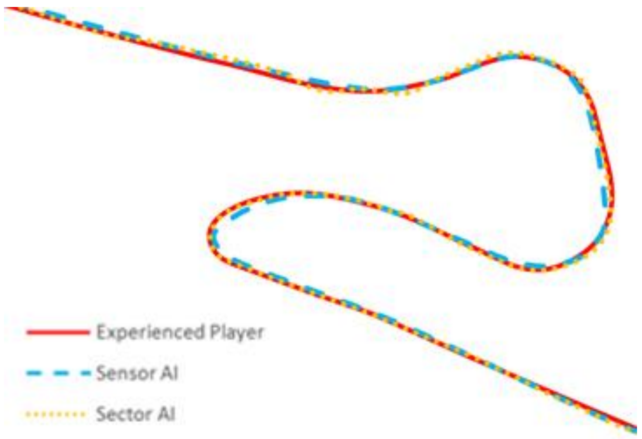


Figure 3.23: A series of corners showing the lines of the player and the AI. From this series of turns, we can see similarities in the lines between the player (red, solid) and the sector AI (yellow, dotted) along the straights, while the sensor based AI (blue, dashed) is able to match the apexes of the player, while also matching the lines in the longer corners.

Despite targeting problems with the sector system for certain parts of the track, the line is believable for the most part in the section covered in figure 3.23. With the sensor AI, it can be seen how much this system stays away from a line that would generally be taken be humans. However, based on how the system finds its line, it is understandable how this is the case.

Although the racing line needs to be believable to convince the player, another important factor for the AI is to also be competitive with their lap times. Lap times can be affected by multiple circumstances, but for this test, they could only be affected by their line and speed. After perfecting the line, developers can start to tweak the speed constraints of the AI. For the sensor based AI, this may require editing the distance equation which determines the speed they aim for, while the sector method would require changes to the target speeds. It is important to only change the target speed rather than rate of acceleration or braking as this could break any sense of immersion, especially if the AI cars are underpowered/ overpowered.

Since both AI methods produced faster lap times than the player, they need to be tweaked slightly to ensure they are competitive. The amount that each system needs to be adjusted is different, with the sensor system requiring a bigger difference as a result of the time gap to the player as compared to the sector method.

3.5 - Requirements for the Hybrid System

Before developing the Hybrid system, the requirements in which need to be met must be identified. These requirements range from set-up, to behaviours for the AI to perform. The requirements will be split into different categories.

3.5.1 - System Set-Up

One of the issues identified while developing the sector system was how long the interfaces took to set up. It should be noted that this was due to not having a system to deal with creating new interfaces in a convenient manner. This is an issue that will be addressed when developing the new system.

One way of achieving this goal is to develop a tool that requires a minimal amount of manual interfaces to be placed down, with more being added via a control panel, allowing the option of manual adjustments being made to the interfaces if required.

3.5.2 - Path finding method

Prior research and development of the two path finding methods provided knowledge to help towards finding a compromise to this problem. As the hybrid system will have an external Unity editor script to aid in setting up the interfaces from the sector method, it must be determined how they should be used. Based on comparative data from the racing lines taken from both sensor and sector systems, it can be seen that the sector method gives better results on the straights, while the sensor system provides a more accurate cornering line. Therefore, it would be beneficial for the AI system to use interfaces for navigating straights, switching over to the sensors when tackling corners. These methods will switch back and forth, providing the best racing line possible.

3.5.3 - Behaviours

The final part of the system that needs to be planned is a set of behaviours. These behaviours can be categorised into certain sets. The identified sets of behaviours for this system at this stage are:

- Race Start
- Racing
- Pit Lane
- Retirement

Race Start – A behaviour set which the AI performs before the race starts. The behaviours range from getting the current state of the race start procedure, to the application of the throttle with variation to reaction times.

Racing – This behaviour set is where the AI should spend majority of its computational life. It is here where the path finding methods will be used. These behaviours take over from the race start as soon as the AI is moving. The behaviours inside this group can split into two further groups, which are:

- Normal
- Attacking

Pit Lane – This will control the decisions the AI makes when it drives through the pit lane. These behaviours are unique as it has a set speed limit for the AI to drive at. The pit lane can be used as a means of punishing the drivers with a penalty, by making then drive through at this reduced speed. The pit lane is also primarily used for the drivers to make quick changes to the car which could give then a winning edge.

Retirement – This behaviour set tries to safely remove an AI car that can no longer continue the race from the track.

3.5.4 - Full List of requirements

The final list of requirements in which the Hybrid system needs to meet are as follows:

- Development of a Unity Editor script to aid in the process of setting up.
  - Setting up the path to use as minimal manual interfaces.
  - Ability to delete and add new interfaces in between the ones placed (known as sub-interfaces.)
  - Sub-interfaces to be automatically positioned between the two interfaces, equal distances between them, with the interface size and rotation changing to gradually transition between the interfaces.
- Implement the two different path finding methods into the hybrid system:
  - Navigating the straights will be completed with the use of the sector based method.
  - AI will be guided through the corners by the sensor method, ensuring that the AI will be able to find apexes at each corner.
  - Collision avoidance will be active in both systems, with the sensors being used to determine when to take avoiding action (via the overtaking line or by other means.)
- To design and produce a set of behaviours in which are able to exhibit those expected for a racer to perform in certain scenarios.
  - Behaviours split into three groups: Race Start, Racing, Pit Lane
  - To use the sensor method for collision avoidance when the AI approaches a slow/stationary vehicle.

Chapter 4 – Developing the Hybrid System

4.1 – Introduction

This section goes through the process taken to realise the artefact. Each section describes the method taken to produce certain features, documenting any issues that may have been identified with the program. Development choices were made as a result of research, and this will be explained.

4.2 – Setting up the track for the `Hybrid System

When setting up the track for use of the sector based system, it quickly became apparent just how long the process took. With this and the project requirements, the total time of setting up the system had to be reduced.

To start this process, a new interface prefab was created. The layout of this prefab was similar to that of the sector method, with waypoints for the different lines. A Unity script was also used to store important variables such as *targetSpeed* and *interfaceNumber*, with new variables to make the development of the Hybrid system easier, such as *subInterfaceNumber, endOfStraight* and *afterSlowCorner.*

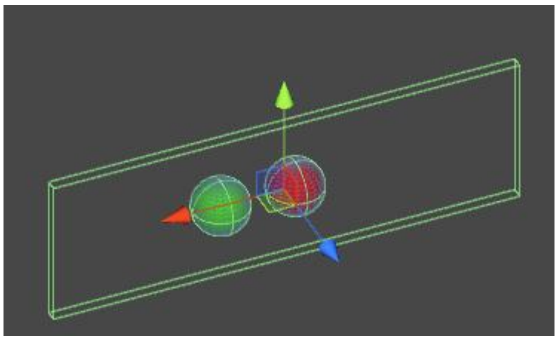


Figure 4.1: The structure of the interface prefab. Each interface contains a racing line node (green sphere), overtake node (red sphere) and trigger box which acts as the interface switch.

After this prefab was created, a game object would be created to assign the system to, allowing the Unity editor script to be accessed. The main goal of this system is to reduce the time taken to place down each interface. This process only requires interfaces in three key areas; *the Start/Finish line*, and at the start and end points of the straights.

By reducing the amount of manually placed interfaces, the amount of time needed to prepare the system decreased. More interfaces can be placed down by selecting the interface they wish to add to, and the number they wish to add. The system creates the requested *sub-interfaces* between the selected interfaces. The placement of the sub-interfaces would be equally spaced out, with the waypoint positions being gradually moved, producing a straight line between the interfaces.

The sub-interfaces are added to the hierarchy in Unity, allowing the user to select any one of them to edit if required. Setting up the track for this new system took approximately three to five hours, greatly reducing the time required to set up the track.

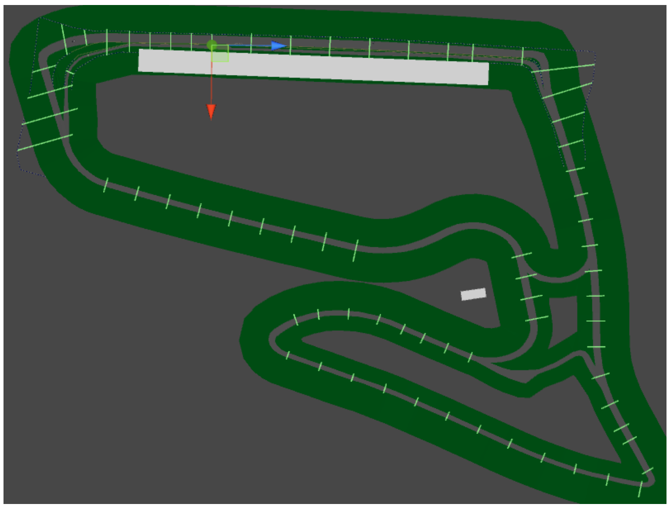


Figure 4.2: The track set up for the Hybrid system. The set up here highlights the lack of interfaces needed at corners, with the choice of switching to the sensor based method while navigating those sections being made.

4.3 – Sector Based System implementation

The first navigation system added to the hybrid system was the sector based system. From chapter 3.2.1, the system proposed by Biasillo (2002) was implemented, with the racing line tested.

4.3.1 – Setting up the system – Comparison

With the Unity Editor tool allowing the user to easily produce a set of navigation points around the track, with the ability to add and remove new interfaces as previously discussed in section 4.2, this section will compare the differences the tool has on setting up the track.

Adding new interfaces to the scene in both systems were different as a result of separate approaches. While the sector system had its environment set up manually, the newer system requires as minimal manual placement of interfaces, with the new tool adding in interfaces at the press of a button.

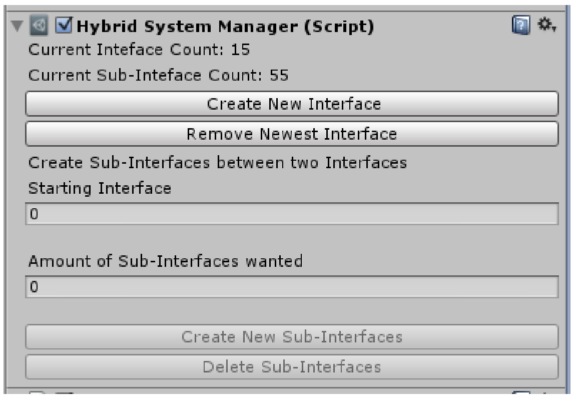
The first version of the sector system also required manually assigned values. This information would determine certain things, such as its position in the interface order. This issue was a direct consequence of dragging in an instance of the prefab to create a new interface. As well as assigning the variables with the information, the user would need to also work on the interface itself, changing its position and size. This would be completed repeatedly until the whole track was set up for use of the system.

With the knowledge of setting up the previous system, a new set-up method needed to be found. Although using the Unity Editor made the total process longer as the tool had to be tried and tested, it did reduce the time taken to repeat the set up seen for the first system. This was the result of automatically assigning each variable when a new interface is created, as the tool is able to keep a record of values.

When the first interface is created, an instance of the prefab is spawned in with default settings. The user can move this instance into position, while making adjustments needed to the waypoints and the overall size of the interface. Once completed, the user can then add the next one in. With one interface now set-up, the tool will use information from the last interface to assign the next one. For the second interface, the position, rotation, waypoint locations and size of the interface will match that of the first one, with the number of the interface assigned to two. Just like the first interface, the user is able to move and alter this new interface to their preference. The second interface should be placed just before the first corner. Theoretically, every odd numbered interface should be placed after a corner, while the even numbered interfaces are placed just before corners; however, this is track dependant.

The straights should be empty at this stage of the setup, with the AI not having enough data. To accommodate this, the tool was designed to add sub-interfaces between two chosen interfaces. The user specifies the interface they wish to create sub-interfaces from, with the number they wish to add. This tool can create up to fifteen sub-interfaces in a single set.

When spawning a set of sub-interfaces, the tool takes into account the sizes, positions and rotations of the two interfaces they place them between. The positions of the sub-interfaces are placed equally apart, with the rotation and size also gradually adjusting from the first selected interface to the second. These sub-interfaces are editable like normal interfaces; however, with the positioning already done, the user will just need to alter variables such as rotation, size and the waypoint positioning. Figure 4.3 shows the interface of the editor developed for setting up the system.


Hybrid System Manager (Script)

Current Inteface Count: 15

Current Sub-Inteface Count: 55

Create Sub-Interfaces between two Interfaces

Starting Interface: 0

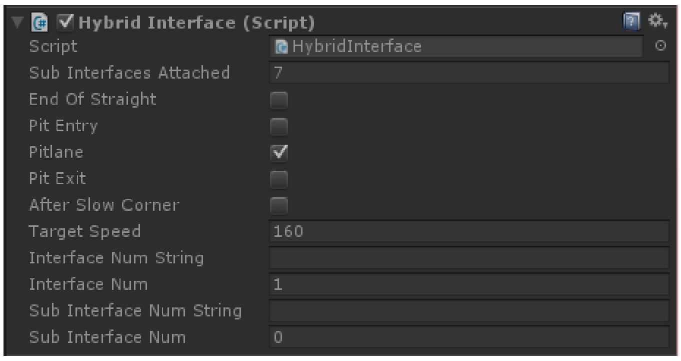
Amount of Sub-Interfaces wanted: 0

Figure 4.3: Interface used for setting up the track.

4.3.2 – Adjusting to the Sub Interfaces

As a result of this new set up method for the hybrid system, adjustments were required for the sector based navigation to be compatible. Without these adjustments, the AI would only aim for the main interfaces. For the AI to determine if it should aim for a sub-interface, system needs to inform it of the presence of these interfaces.

This was achieved with a variable, known as *subInterfacesAttached*, which was created at the beginning of development alongside the interface prefab. This variable is assigned a number when any sub-interfaces are instantiated via the tool. This number, which is also the number of sub interfaces the tool creates, is stored within the interface selected by the user. If an interface has its sub-interfaces removed, the value is reset to zero. Figure 4.4 shows the properties of the first interface, showing how number of sub-interfaces are represented in the interface script.



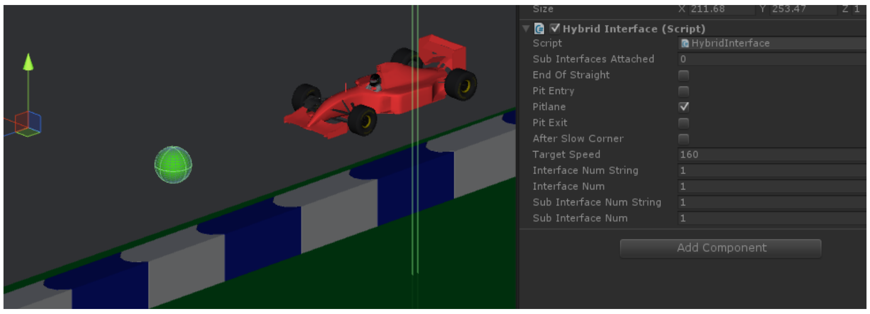
Hybrid Interface (Script)

Script	HybridInterface
Sub Interfaces Attached	7
End Of Straight	☐
Pit Entry	☐
Pitlane	✓
Pit Exit	☐
After Slow Corner	☐
Target Speed	160
Interface Num String	
Interface Num	1
Sub Interface Num String	
Sub Interface Num	0

Figure 4.4: Properties of interface one.

When the AI targets this interface, it will check the sub-interface count attached to it, assigning it to its own variable for later reference. Along with its interface counter, the AI also has a counter for sub-interfaces. This counter is only increased if the value of the sub-interface the AI is aiming towards is less than the value stored in *subInterfaceAttachedCount*. If *nextSubInterfaceNum* equals the value of this variable when the AI goes to the next sub-interface, then *nextSubInterfaceNum* will be reset, while nextInterfaceCount moves onto the next interface.

One issue that would be encountered is when the AI would switch between an interfaces to a sub-interface. As previously noted, the AI is made aware of the presence of any sub-interfaces when they target an interface. However, as the sub-interfaces are also instantiated from the same interface prefab as the normal interfaces, the variable used to declare how many sub-interfaces there are is assigned a value of zero. To prevent this, the code in which assigns the value from the script needs to placed where it is only called when *nextSubInterfaceNum* equals to *subInterfaceAttachedCount*. Figure 4.5 shows the AI targeting a sub-interface, as the properties confirm, while using the sector based method.



Size	X 211.68	Y 253.47	Z 1

Hybrid Interface (Script) ✓

Script	HybridInterface
Sub Interfaces Attached	0
End Of Straight	☐
Pit Entry	☐
Pitlane	✓
Pit Exit	☐
After Slow Corner	☐
Target Speed	160
Interface Num String	1
Interface Num	1
Sub Interface Num String	1
Sub Interface Num	1

Figure 4.5: AI targeting the racing line node in a sub-interface.

4.4 – Sensor Based System implementation

Transferring the sensor based system over and ensuring it co-exists alongside the sector based AI were the next steps of development. While transferring this system in, the sector based system would be disabled.

4.4.1 – Transferring the Sensor system over

Having produced a working navigation system that uses information gathered by sensors similar to the method by Butz, et al (2011), with a slight modification as a result of the system by Wang & Lin (2010), the process of transferring this system into the hybrid could now begin.

A decision was made to contain both the sector and sensor based systems within the same Unity C# script. This decision was made on based on the target of decreasing the time developers would need to spend in order to set up the system. This would also allow all functions used in both systems to be made available to the script without needing to cross reference external scripts, meaning less set up time. However, it is important to keep the functions of the two systems separate, ensuring that the correct behaviour is performed. The majority of these functions were able to be transferred without problems, while some may have required a name change, forcing adjustments to the call functions. One major change that had to take place was the processing of the data in order to get the car moving.

Since both systems use the physics engine to move the AI vehicle forward and to steer, the behaviour for this should be kept in the same function. Although both systems move the car in the same way, they determine how much they need to move the AI by via different methods, requiring different inputs. To ensure that the right method of moving the vehicle, the two different methods were separated by an *if statement* within the *ApplyMovement* function. The if statement would determine which system should be in control by the value of *cornering*, with a value of true meaning that the sensor system should be in use. The value would be assigned to true for this process, until the system was working 100% just like the standalone version.

4.4.2 – Setting up the track

The sensor part of the navigation needed to have the track set up accordingly to accommodate its path finding. However, unlike the sector method, the procedure of setting up the track for the sensor system had not changed. An extra game object would need to be created so that both inside facing and outside face track edges could be used, as previously added to the standalone system.

4.4.3 – Ensuring the system was ready

As the track set up time remained unchanged when transferring the system, the last stage of this process was to ensure that the system was working. When this side of the system is being tested, the results of the behaviour would be compared to that of the previously developed system.

When activating the program, the AI would go off on its own, navigating the track and handling corners. Observations showed that the AI took straights like the standalone version would, sticking to the middle of the track. Cornering also proved successful, as the AI was able to find the apex on a majority of the occasions.

The final test of this system was to ensure that the system wouldn’t control the car when the value was switched to false. Changing this variable was done via the editor panel while having the AI selected in the hierarchy. To change this variable via the inspector during runtime it would need to be made public. When this variable is changed from true to false, the system switches over to the sector based system, confirming that when the value is false for *cornering*, the sensor system is inactive. Reversing the change would enable the system.

4.5 – Getting the two systems to co-exist

With both methods transferred into the hybrid system, it was time to get the AI to use both of them at the appropriate times.

4.5.1 – Switching to the Sensor system

The interfaces already contains a variable that the AI can extract, informing them if a corner is after it, which the system can also use to switch to the sensor based method. The variable *cornering*, is set active when the AI hits an interface with the variable *endOfStraight* set as true. As this variable is only active before a corner starts, the AI should know when the sensor system should be used. The interface can change the variable within the AI script by using the collision reference to access the AI script.

4.5.2 – Switching back to the Sector system

Switching back to the sector based method was programmed to occur when the AI car would enter a new straight. This switch takes place once one of two scenarios are completed. The first scenario is where the middle sensor (going directly forward) makes contact with the appropriate waypoint on the next interface. The second will switch back to the sector system if the vehicle has hit another interface, acting as a failsafe.

4.6 – Introducing height variation to the track

With the hybrid system now being able to use both methods, the next task was to see how the AI would behave when navigating the track with height variance. It’s important for the AI to have no problems navigating a track in which isn’t constantly flat, as most tracks will present this issue.

4.6.1 – Adding height to the track

Height was added to the track via the program Autodesk 3DS Max, with multiple areas of the track having their Z values increased. Figures 46A and 4.6B show the differences in the track height before and after the changes were made, with the screenshots taken within Unity.

Figure 4.6A: A side shot of the track in Unity without height.

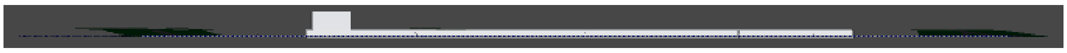


Figure 4.6B: A side shot of the track in Unity with height.

Before testing began, two more steps were completed. The first step involved the track parameter objects, which is used by the sensor system to get the best possible path. These objects needed to have their height increased to deal with not only the highest point on the new version of the track, but also the possibility of having the sensor target being even higher as a result of moving up hill. This process was also completed within 3DS Max, with the resulting collision mesh shown in Figure 4.7.

The last change that was required involved the interfaces in which were present in the scene. In order to compensate the new height in certain areas of the track, the trigger zones would need to be made taller. This would be done via the Unity editor, increasing the Y value of the triggers size vector.

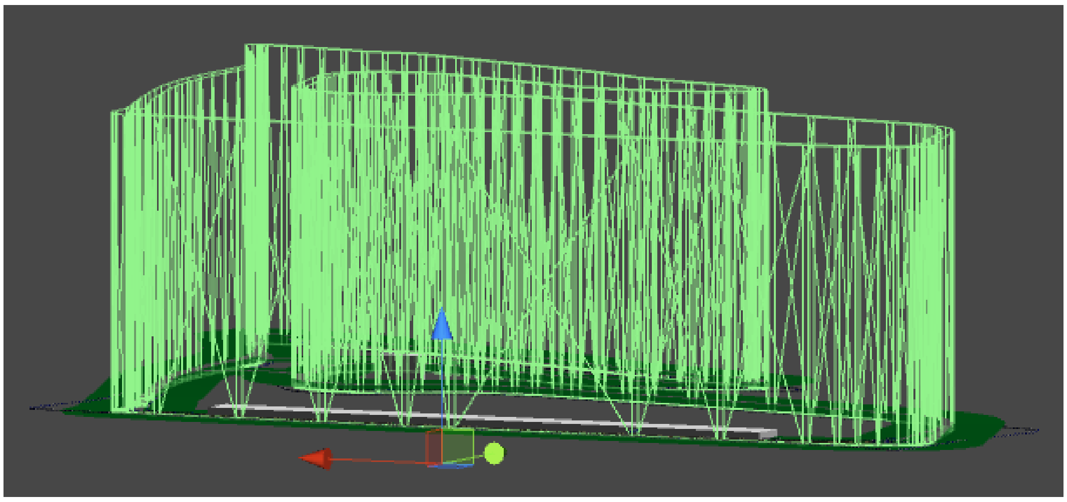


Figure 4.7: The altered track limits to accommodate the height.

4.6.2 – Testing the AI on the new track version

With the new track applied to the program and the relevant modifications made, the AI could then be allowed to navigate the new environment. Problems with the system were not met until the vehicle had entered the second straight. From this point, the AI would begin ascending up the track, beginning to rotate downwards. This behaviour was to be expected as the AI in the sector based method is using a 3D location to aim its vehicle. This meant the sensor based AI had to be adjusted to navigate an elevated part of the track.

4.6.3 – Making adjustments

After understanding the reason for the issue, adjustments were then made. As mentioned, the problem involved the car rotating down towards the waypoint as the car starts ascending on the track. The cause of this issue was the AI aiming at a 3D location rather than a 2D location. This would be altered for the AI to be able to move along varied heights, without the need of having to adjust the waypoint locations at this stage.

Before adjustments can be made, the coordinates the AI must ignore needed to be determined. Finding this coordinate involved the process of moving a waypoint in the three axes. The axis which would affect the height of the waypoint is the Y axis. With this knowledge, the adjustment of the sector systems targeting method can be made.

To ensure the AI doesn’t aim down when on an incline, the targeting system will need to target the waypoint, excluding the Y coordinate. Leaving this coordinate empty will set the value to zero, meaning this behaviour will still be performed, requiring a value to be assigned. The solution was to assign the Y coordinate of the car to its target. This allows the car to move up or down in height, without suffering any consequences.

The sensor based system was also tested here, with results at this stage showing that the system worked fine, with no further adjustments required.

4.7 – Collision Avoidance

After getting both systems to work together to navigate around the track, along with being adapted to height variation on the track, the next development step was implementing collision avoidance. It was identified that there would be multiple forms of collision avoidance which needed to be implemented.

4.7.1 – Setting up the detection system

When preparing to implement collision avoidance, the first task was to determine the best possible way to collect information of the AIs surroundings. When developing the sensor based system, the realisation of how useful the sensors were for collecting data. As a result of this, more sensors would be implemented around the car, allowing the AI to collect data of the environment around them.

Four different groups were identified during the set up process of the sensors: front, back, left and right. These sensors would face out in the direction of the group they are placed in, meaning that the AI is able to collect data for all four sides of the vehicle. At this stage, it was also important to produce a second spawn point for the sensors located at the rear of the vehicle. This would be the spawn position for the sensors in the rear group, as well as the rear sensors in the left and right groups. The front spawn point was already present, being predominately used in the sensor navigation method.

With the sensors ready, an attempt to reduce the processing power these sensors would take up when active was explored. One technique was to use trigger boxes to activate a group of sensors, meaning the sensors could remain inactive unless their corresponding trigger detected an opponent. If a trigger detects an opponent while another sensor set is active, the AI will disable that first active set to ensure that the only set active is that of the latest active trigger. Figure 4.8 shows a diagram of how the triggers are set up around the car, showing an example of when the sensors are activated.

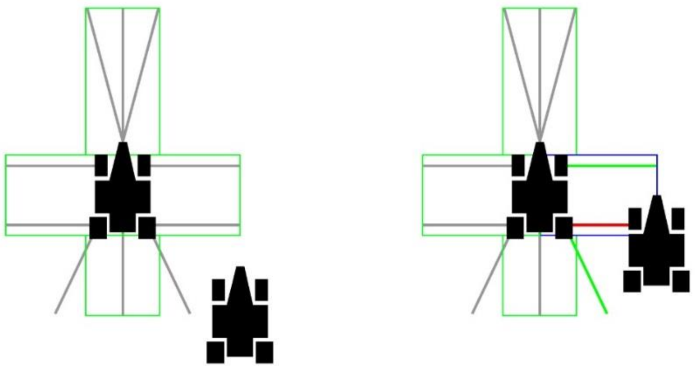


Figure 4.8: Diagram showing the setup of the trigger boxes.

4.7.2 – Overtaking

The first type of collision avoidance is one of the most important behaviours in a racing game, allowing the AI to make an overtaking manoeuvre. This collision avoidance is triggered when the AI approaches an opponent and getting within a certain distance, which varies depending on the speeds of both the AI and its target. This avoidance behaviour will make use of the second (overtaking) line within the sector method.

With the trigger boxes set up to determine when each set of sensors should be activated; the decision on which sensors should trigger the behaviour had been planned. With the car that will attempt the overtake being the chasing vehicle, the front sensors would provide the necessary information required. This is because the AI would be able to tell if the distance between both of them is getting smaller, and as a result, know if it's going faster. If they were side by side, the chances are that one of the cars has started to overtake. This behaviour would take place while on the straights, as the interfaces offer the second path. While tackling corners, a separate behaviour would be implemented to make the AI steer away from each other to avoid colliding with each other (see section 4.7.4.)

Getting the AI to follow the overtake line rather than the racing line would require information for the AI to judge if a move can be made. As the targeting system was able to change between the different lines (manually) in the standalone version of the sector based system after changing the value of the *overtaking* variable, it means that the same can be done for the hybrid system.

The AI can use the overtaking lines in the sub-interfaces, as well as switch between both lines at any time during the lap. One important decision making task that needed to happen was to get the AI to decide when to overtake, as well as to decide when it should return to the racing line. Once the distance of the AI to the car in front is less than the distance needed, the AI knows it can then overtake, and will proceed to target the overtaking line. It does this by changing the value of overtaking in the *HybridNPC* script to true. Figure 4.9 shows an AI car (red) overtaking another car (purple) by targeting the overtaking node in a sub interface.

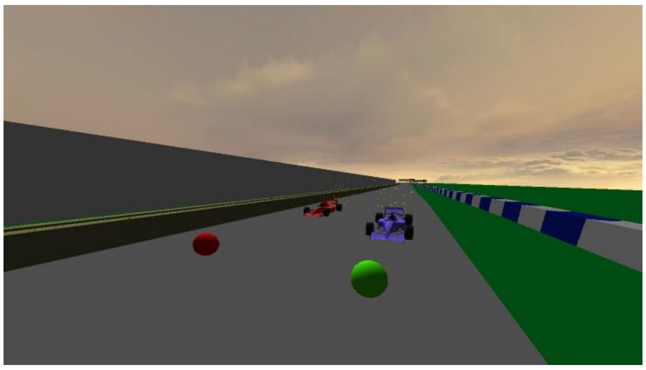


Figure 4.9: An AI car attempting to overtake another via the overtake line.

This value change will also cause the AI to trigger a function called *TargetChange*, which changes the AIs target node, based on the next interface. This behaviour allows the AI to switch between the two lines. This also happens when the value of *overtaking* is switched back to false, with the AI switching back to the racing line. Switching back to the racing line was the last part of the behaviour in which the AI will complete.

Decisions based on when the AI should return to the racing line made use of the rear sensors. These sensors allow the AI to know if the car they were trying to overtake is currently behind them. When the AI detects that the opponent behind them, whether it be as a result of the left, right or middle sensors, the AI will decide that it's overtaking manoeuvre is complete, changing the value of *overtaking* back to false. By allowing any of the back sensors to detect this scenario helps to prevent the AI from staying in this behaviour when exiting corners, where the AI will detect the other car mid corner, prompting the change in the *overtaking* value.

4.7.3 – Slow/Stationary object avoidance

The next collision avoidance behaviour implemented is used to avoid an obstacle that is travelling considerably slower than the AI or not moving at all. This can be due to an AI opponent retiring or having crashed out. Unlike the previous behaviour, this avoidance does not use a pre-determined line to navigate around, but uses the sensor system instead. The reason for using the sensor system is due the object triggering the behaviour potentially blocking both lines, which the AI would have to navigate around. This isn't a problem when the AI is overtaking another car as it would be safe to assume that they would be travelling in the same direction, not blocking both lines.

To make this behaviour seem more believable, the front sensor trigger box would be made considerably longer. The concept behind this decision is based on driver vision. The AI needs a considerable amount of time to see the problem, and to anticipate making adjustments to its path, with the current size of the trigger box being deemed too small. It also means the AI is able to see further down the track, similar to what real world drivers would be able to do, rather than a few metres in front of them. Figure 4.10 shows the new size of the front trigger box, giving the AI a greater amount of time and distance to react to the obstacle.

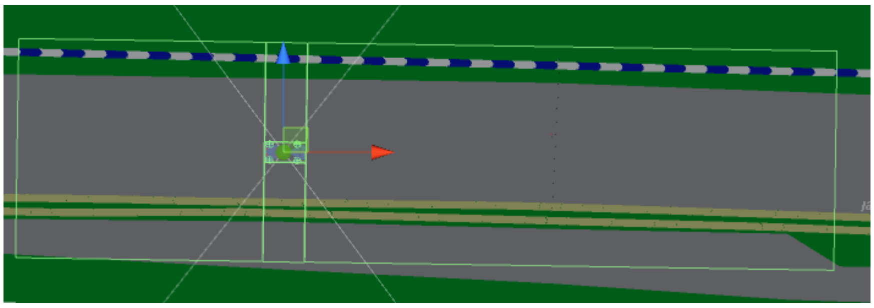


Figure 4.10: The new size of the front trigger box on the AI. The increased length of the trigger box allowed the AI to detect objects at a further distance, crucial for collision avoidance.

Once the front sensor is activated, the ray aiming straight forward is used to detect any slow moving objects. If an object is detected, the ray collects the *currentSpeed* value of the object and determines if it is travelling considerably slower than itself. If AI has determined that this scenario is true, the navigation system switches over to the sensor based system.

With a new scenario added to switch control of the vehicles navigation, a switch in the form of an if statement was created. This if statement is placed among the other if statements responsible for deciding which system should be in control. Fortunately, due to the way in which the if statements were organised, an altered if statement that checks for the value of this variable can be used. This if statement is placed within the *HybridNPC* script, and bases its value on that of the *sensorSteering* variable, which is changed within the *CollisionAvoidanceV3* script. Figure 4.11 shows the AI using the sensor system to make its way past a stationary vehicle that has stopped in a position that blocks a portion of the track, while still using the sensors to detect any new objects that may need to be avoided.

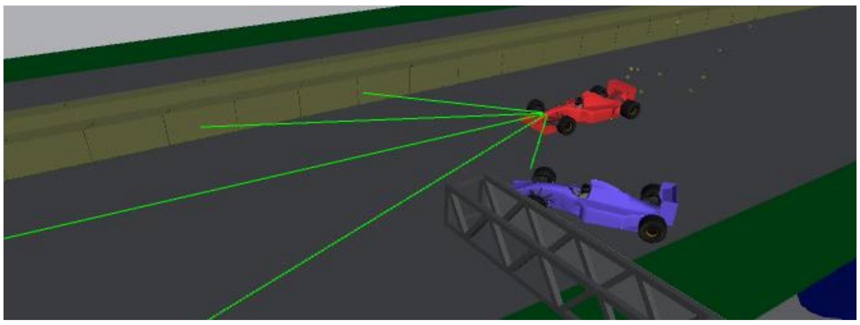


Figure 4.11: The AI using sensors to navigate around the stationary car.

This new behaviour also requires a new transition event to give the sector system control once the object has been passed. While the system could be switched back to the sector base system after the AI navigates the next corner, the aim is to switch back before the corner (if possible), allowing it to find a better line. Implementing the sensors used to check around the car for opponents was how this transition event was done. Primarily using the sensors at the back, the decision to revert back to the sector system it will assume that the line is now clear to re-join. At this stage, the *sensorSteering* variable is assigned as false, meaning the sector based system is now navigating for the AI.

4.7.4 – Side Avoidance

The last avoidance behaviour coded for the AI to perform is available in the event that the AI found itself getting too close to another AI on its side. The aim is to adjust the targeting system of the hybrid system so the AI would aim slightly to the left or right of its current target. The sensors the AI would use to determine if this behaviour needs to be performed would be the side sensors.

As these sensors are only activated when a car has been detected in the side triggers, these behaviours shouldn’t be enabled unless the AI is side by side of a vehicle. To get this behaviour to work, a separate game object which the AI can relocate in order to help them toward the position would be needed. However, the sensor system already uses a similar method of targeting; resulting in the game object from this part of the system being used.

Adapting the AI to aim for this object required essential information to move the target; the side the opponent is, the distance between the two vehicles and the coordinates of its current target. These values will be used to determine how much the target needs to be altered by. For this process, a second AI car is created. The second car will be moved left and right to see how much of a difference the system is having on the targeting system. The behaviour to alter the target would be triggered if the distance would be below a certain value, and as a response, the AI would move in the opposite direction. The AI would continue this behaviour until the distance was back to its minimum.

This behaviour would be triggered with the use of a Boolean variable, which would be true if the distance was less than the minimum. This variable then activates a function which goes through the process of moving the target object based on the position of the current target, and move it ten units on the x axis in either direction. However, in the event that the AI would go side by side into a corner, then the target object would be placed as normal with alterations taking place afterwards.

Chapter 5 – Artefact Testing

5.1 – Introduction

To ensure that the artefact is fit for purpose, a number of tests in specific scenarios will be performed. These tests will focus on certain aspects of the system, with them being conducted numerous times to get the best results and to find any oddities that may exist.

5.2 – Test Cases

This section explains how the system will be tested to determine it is fit for purpose, and how beneficial the system is.

5.2.1 – Racing line

The first test involves recording the line taken by the AI. Recording this line will take place on lap two, ensuring equal and fair results when comparing the line against those generated by the sensor and sector based methods. This line will be drawn five times, each being taken into consideration before choosing the best line to compare with the other methods. To compare the lap times, a lap that is judged to be within the middle of the range would be chosen, and used to compare against the lap times of the two previous methods.

In order to make this comparison fair, the hybrid AI will be tested under the same conditions in which the lines from the sector and sensor methods were also under. This would ensure that the comparison of the three systems is fair and transparent. The main differences between the hybrid testing compared to the testing conditions of the other systems are that the hybrid method is to be tested on the track which includes height variance.

It is expected that lines produced from this test is similar to those of a player in terms of behaviours, with the positioning down the straights and corner entries being similar.

5.2.2 – Collision Avoidance (Stationary object)

The next test case will set out to see how effective the collision avoidance is when it detects a stationary object. This system will be tested with two different scenarios. It should be noted that each of the tests conducted below will be under the same testing conditions. The only exception to the conditions that will be different for each test is the direction in which the stationary car is rotated.

5.2.2.1 – Test #1: Car parked on the track

The first test the system performs against the stationary target will be where the opposition is parked along the racing line, facing the first corner. This will simulate the car having to avoid an opponent which has retired from the race, having not been able to make it off the track. Due to the width of the car, the AI isn't expected to wonder too far off the racing line, and should return to it once it has cleared the car. Each test will be under the same conditions, with the car facing towards the first corner, in the same position on the racing line while the AI starts from the same location with the same starting speed.

5.2.2.2 – Test #2: Car facing the wall on the track

The second test looks at how the AI behaves when coming up to a stationary car in the same place, but taking up more of the track. By having the stationary car facing the barriers, it gives the scenario that a car has crashed out, and the AI needs to make a bigger change in the line it takes to get past. Again, the AI is expected to return back to the line once passed. Just like the previously set out test, the conditions of these five tests will be the same, except the car stationary car will be face in the pit wall while resting on the racing line.

5.2.3 – Collision Avoidance (Slow object)

Testing how well the collision avoidance behaves when overtaking a slow moving vehicle will be explored next. As this part was developed alongside avoidance of stationary vehicles, the behaviour exhibited should be similar. This test will be conducted five times, with the speed of the opponent being increased by increments of five in each test. The expected result of each test is that the AI will move off line via the sensor based method, and return after clearing the slower car. An expected variance of the results is expected, as the AI should have to make its passing move later as the tests progress, as well as overtaking taking slightly longer as the speed is increased.

It is important to test different conditions for this avoidance feature in order to understand what speeds the AI will decide to use the sensor method in to pass an opponent. As a result of this, the test was conducted five times, with the speed of the slower vehicle increasing each time. From this test, we should see that the AI reaches the slower vehicle, switching to the sensor method in order to get pass. As the speed of the slower vehicle is increased with every test, the point in which the AI switches between methods should get further down the track as the testing progresses.

5.2.4 – Start Procedure

The fourth test case examines the behaviour of the AI during the start procedure. Each test will examine the behaviours followed by the set of AI cars. Each car will be placed on the starting grid, and will operate to see how successful they are at moving once the race starts. The grid size will start with four cars, with each following test increasing the grid size by two. The main points of concern in this test is to find out if any AI cars are capable of false starting and how often does the problem occur, if at all.

False starting (also known as a jump start) happens prior to the race officially starting, which results in the competitor moving before the lights turn green, signalling the start of the race. This gives the competitor in question an unfair advantage, as is usually given a form of penalty. This test is important, as it determines whether or not the AI fairly moves of the start grid. It should be noted that each test is conducted under the same conditions, with the exception of the number the AI that will be present. Each test was recorded, with the results recorded based on the number of cars that move before the green lights are activated. This test is designed to see if the AI is capable of jump starting, with the expected result returning as false, indicating that no AI was able to perform this behaviour.

5.2.5 – One lap racing

The fifth test will have the AI race against each other for one lap. The behaviours shown by the AI cars will be noted along these short races. The testing will be done over the course of three races, with any unexpected behaviours being recorded. Each race will start with a different amount of racers, which would allow the chance to see the variance in behaviours.

5.3 – Test Results

This section will look at the results from the tests planned and conducted from section 5.2, and will discuss what they mean in terms of overall performance. These results are critical, as they will be used for the overall evaluation at the end of this chapter.

5.3.1 – Comparison of Racing Lines

The first test was designed to see how believable the line the AI takes is compared to the previously implemented systems. The test recorded five lap times, along with the own lines, which can be used to compare against the sector and sensor methods. If the range of lap times is between or close those of the sensor or sector methods, then the speed and difficulty of the system can be discussed. Figure 5.1 below shows the results of the lap times of the five tests:

Test	Lap Time	Line used to compare?
1	1 min 06.686 secs	✓
2	1 min 07.045 secs	✕
3	1 min 06.279 secs	✕
4	1 min 06.265 secs	✕
5	1 min 07.653 secs	✕

Figure 5.1: Results from the five testing laps.

It quickly became apparent that the lap times the AI was managing to produce were quicker than both of the two previous systems. These lap times had a variance close to 1.4 seconds, despite the slowest time having experienced a slight crash. By omitting this result, we can see that this variance decreases to 0.78 seconds. Because of the crash in the fifth test, the line in this test cannot be used. With the remaining four lines, two have similar lap time, which are considerably faster that the next fastest lap. As the result of test one being in the middle of tests two and four's lap times, this line was selected for the comparison. Figure 5.2 shows the line from test one being drawn against the lines from the sector and sensor methods, along with the experienced player.

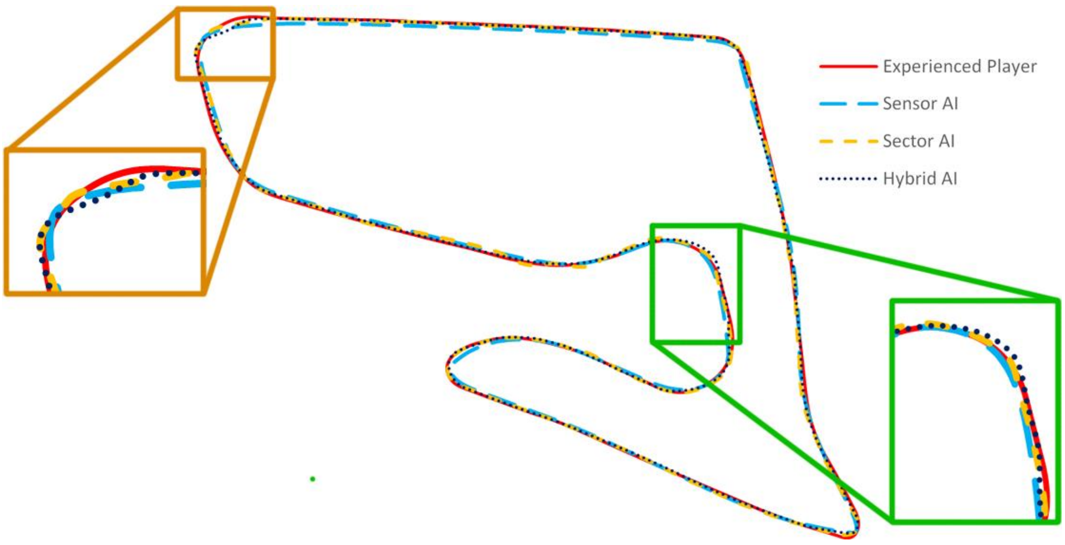


Figure 5.2: Comparison of lines from the AI systems and a player. It can be seen that the line taken by the new hybrid method (navy, dotted) is similar to that taken by the experienced player (red, solid), with a few odd behaviours, in which two are highlighted via the boxes.

From the lines drawn in figure 5.2, it can be seen that for the majority of the track, the hybrid system was able to incorporate the two systems appropriately to produce a more believable line. A few characteristics can be seen from these lines. Firstly, the line taken on the straights matches that of the sector based AI, which closely resembles the line taken by the experienced player from section 3.4. Also, looking at the corners, the sensor method also allows the hybrid system to find the apexes of the corner much better, with certain corners mirroring that of the player.

Unfortunately, the AI does have an issue when exiting corners. Although the severity of the problem is dependent on several factors such as the corner tightness, speed and placement of the interfaces, with the problem visible in three areas. Figure 5.3 shows the lines at the final corner which demonstrates this issue.

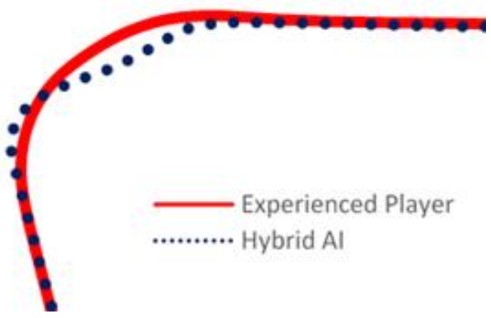


Figure 5.3: The last corner showing an oddity of the Hybrid system. The line drawn by the Hybrid method (dotted line) shows three turning actions to navigate the corner, compared to the line taken by the player (solid line) using just one. As well as this, the line of the hybrid method also shows the AI turning later, which is a result of the interface.

The problem seen at this corner is that the AI seems has trouble switching back the sector based system before hitting the interface. Back during the development phase, the system was coded to search for the most appropriate waypoint at the next interface, triggering a switch back to the sector method. However, if the AI is unable to find a waypoint, it continues to search for an appropriate line until it hits the next interface. This is happening due to the ray cast from the sensor method not being able to find the waypoint as a result of the collision mesh of the waypoint being assigned as a trigger. This is why the hybrid system produces an odd line at this corner, as it steers from a different line rather to the racing line as the navigation method changes.

5.3.2 – Results of Collision Avoidance (Stationary)

The second phase of testing analyses the collision avoidance of the AI against stationary opponents. This test was conducted to ensure that the collision avoidance was able to detect opponents which have stopped on the track. Due to the starting position of the car, the starting speed was set to 30. This test was set out into two separate testing environments, with the results of each one displayed in the tables below:

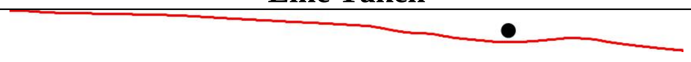
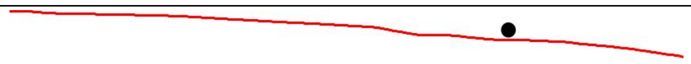
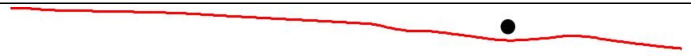
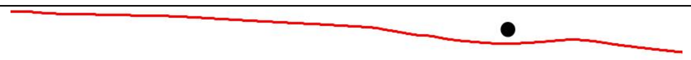
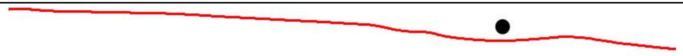
Test 1: Car parked		
Test	Line Taken	Contact?
1		×
2		×
3		×
4		×
5		×

Figure 5.4: The results of the first testing phase. The dot shows the location of the vehicle the AI was attempting to avoid.

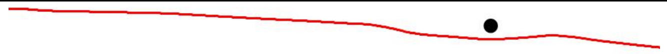
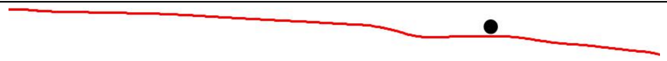
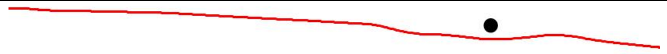
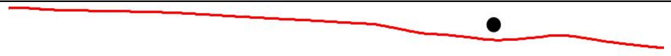
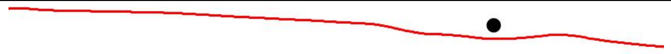
Test 2: Car facing wall		
Test	Line Taken	Contact?
1		×
2		×
3		×
4		×
5		×

Figure 5.5: The results of the second testing phase. The dot shows the location of the vehicle the AI was attempting to avoid.

Results from the tests came back mostly positive. Each of the ten AI cars successfully managed to avoid contact with the stationary vehicles. Despite this success, only eight of these provided the expected results. The two odd results successfully avoided the stationary vehicles but would not return to the racing line. It was at this point in which it was discovered that the AI would not always be able to get the information from the interfaces, resulting in behaviours seen in these two cases.

The lines taken by the AI in both test sets show that the system was able to avoid the obstacles effectively. The varied paths also suggest that they are not using the overtake line, as the different tests showed that the AI would react at varied times. Surprisingly, as figure 5-6 shows, the AI in both test cases used the same off line distance to avoid the cars, despite the first test case having less width to avoid.

Despite the two tests that provided an error, the remaining eight tests showed that the AI was able to return to the racing line, braking for the next corner. Figure 5.6 shows the lines from the fifth tests in both cases, with test case 1 being represented by the red line while the dark blue line represents test case 2.

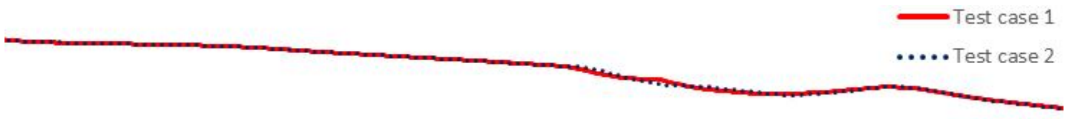


Figure 5.6: Comparison of lines from stationary collision avoidance testing. From this comparison, it can be seen that both AI moved out by the same amount in order to get passed the stationary vehicle, despite test case 2 (dark blue, dotted) having more width to avoid than test case 1 (red, solid.)

5.3.3 – Results of Collision Avoidance (Slow moving)

The collision avoidance was now tested with the opponent moving slowly. This test should provide similar results seen in the first test in section 5.3.2, with the avoidance taking place at different parts of the straight. The results of the five tests can be seen below in figure 5.7:

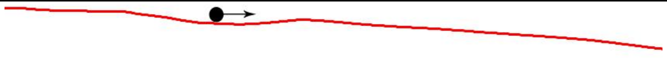
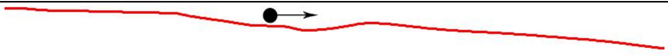
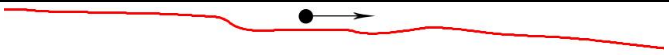
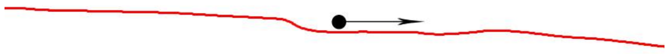
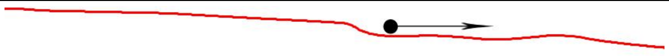
Test	Opponent Speed	Line Taken	Contact ?
1	25		×
2	30		×
3	35		×
4	40		×
5	45		×

Figure 5.7: Results for testing of the collision avoidance against slow car. The dot shows the location of the vehicle the AI was attempting to avoid when it was detected, with the arrow showing the direction of travel, with the lengths signifying the distance travelled before the AI switched back over to the racing line / sector method.

These tests proved to be successful, with the AI cars managing to avoid the slower moving AI. One surprise that happened through the later tests was how the AI passed the AI. The first couple of tests provided expected results, with the cars approaching the slower car, knowing that it would need to move off line and make a pass. The properties showed that this was done with the sensor steering, which was expected.

The third test produced an unexpected behaviour, as the AI would use the overtaking line once in range of the slower car. However, as the car moved out to overtake, the speed was suddenly greater enough to warrant a switch to the sensor avoidance. This test showed that the decision making of the AI was always changing, knowing when to use the overtake line to get past an opponent.

However, the last two tests did not use the sensor avoidance. As the speed of the slower AI was travelling too fast for the sensor avoidance system to be activated, the AI made use the overtaking line to get past. Despite this, the AI was still able to make it through without any collisions. Figure 5.8 shows the lines of the AI in the 5 tests on the start straight, showing the different locations they started their avoidance.

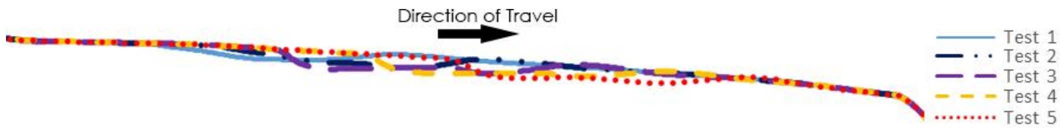


Figure 5.8: The lines taken by the AI to avoid the slower vehicles. From these lines, it can be seen that the sensor method moves the AI outwards less than using the overtake line via the sector method. The lines for test 1 (light blue) and test 2 (dark blue) shows that the AI wouldn't move as wide as the AI in tests 4 (yellow) and 5 (red.) This diagram also shows how the AI in each test moved out at different times down the straight, with the distance down the straight increasing at every test.

5.3.4 – Results of start procedure

The start procedure is an important aspect of a racing game, as it aims to simulate the procedure you see in real life racing. By testing this part of the hybrid system, it ensures that any problems with the system are found. The main issue that is being looked out for in this test is any jump starting AI.

Test	AIs starting	Jump starters?	Number of Jump starters
1	4	✕	0
2	6	✕	0
3	8	✕	0
4	10	✕	0
5	12	✕	0

Figure 5.9: Results of the start procedure testing, showing the number of AI that jumped the start during the race start procedure (if any.)

Each test provided successful results for the behaviours of the AI at the start of the race. Testing this procedure was due to issues experienced when testing with multiple AI cars on the track. The different results from those experienced before can be down to a few different settings.

One of the key differences made to the AI in this test were the cameras attached to the cars. During development, the AI included a camera object which allowed for observation of how the AI behaves. Each camera has several components, with one of them being an audio listener. Problems occur when more than one audio listener are present. One example of this is trigger detection zones, which wouldn't detect the game object if multiple audio listeners are present in the scene. For testing purposes, the cameras and audio listeners in each AI car was disabled, conditions you would expect to see in a racing game.

5.3.5 –Racing over one lap

The final test observes a race containing a number of AI for one lap. This test aims to find how the AI behaved in races, noting down any odd behaviours that were shown by any of the cars.

Test	AIs starting	Odd Behaviours?	List of odd behaviours
1	2	✕	N/A
2	4	✓	AI #3 was able to travel at top speed for the whole lap, resulting in the car sliding off track constantly. AI #1 was able to cut part of the track. All AI cars were bumping into each other constantly.
3	6	✓	Test Aborted

Figure 5.10: Results of the racing tests.

Testing the race craft of the AI provided very mixed results. During this testing phase, a major issue regarding the collision avoidance system was discovered, which lead to the last test being aborted.

With the first test, both of the AI cars sped off their starting positions towards the first turn. During this time, the first AI stuck to the racing line, with the second AI sticking to the overtake line. At the first corner, the first AI would cut across the path of the second AI, with no collisions taking place. From here, the AI cars were spread out, with no more action taking place.

The second test began to highlight a few problems with the avoidance system. All of the AI except the first AI would try to take the overtaking line from the start. This led to collisions, which realised a design flaw of the collision avoidance. The AI is unable to effectively race in a pack. Other notable behaviours included one of the cars constantly going full speed, while another was able to cut a section of the track.

The third test increased the grid size by another two cars. Unfortunately, this test was aborted early into testing due to complications. The first issue was the performance of the program. The reduced frame rate impacted the performance of the AI from the start, as the cars were slow off the line. The first corner was where the test was aborted, as the frame rate made the racing between the cars worse, with more bumping being seen, and cars becoming immobile as they crashed out.

These tests provided positive and negative feedback on how the AI races against each other. While suggesting the AI can race without major problems when one on one with an opponent, it cannot compete with multiple cars in close proximity of each other. This will be an issue for the start of a race where the cars are all within a metres of each other.

5.4 – Known Issues with the system

During development, a few issues were encountered. Some issues we left into the system, with efforts taken to fix or reduce any impact that they may have on the overall artefact. However, some issues were not fixable, and although not enabled in the working version, they are still included.

5.4.1 –Nose diving Cars

With the switch made to the height varied track, down force would be needed to be applied to the cars to ensure they stayed on the track surface. This created an issue as the constant collisions between the car and track would send the car nose-diving towards the track. After reworking the down force calculations as well as adjusting the gravity applied to the car, the effect of the nose diving was reduced, with cars being able to eventually get back into a position where they can drive towards the next waypoint.

5.4.2 – Pit lane issues

The pit lane behaviour was one of the last features to be worked on, and was added in once collision avoidance was implemented. While being a requirement for the hybrid system, the pit lane behaviour was ultimately omitted from the final version due to several issues.

The first issue was due to a conflict with the collision avoidance. This issue wasn’t realised in development as the testing involved only one AI. There was a total disregard of how the AI would behave when attempting to pit while racing for position with another car, rather than the coded scenario that the AI could encounter, which also had its own issues.

The coded behaviour should’ve performed a specific behaviour in certain scenarios. The AI car is meant to skip the pit lane for another lap if it detected a car in front is also pitting on the same lap. This car would then go on to pit the next lap. The expected result was not the result realised, as it wouldn’t enter the pits on the following lap. This behaviour also led to another unwanted behaviour, where the AI would speed up while entering the pits, before deciding to go back onto the race track. This behaviour would cause the AI to crash and immobilizing itself, making it unable to continue.

5.5 – System Evaluation

The hybrid system currently needs more work to it to ensure a high level of believability. However, the system works well in the areas it was primarily designed for. Developers will be able to create believable lines with the hybrid system, despite some areas requiring a bit of work. Using the sensor based method to handle part of the collision avoidance also proved to be effective, as well as the AI behaviour when racing with another opponent. Figure 5.11 shows the lines of the three AI systems. From this figure, the hybrid system can be seen to have produced a line that uses the same line as the sector based method, while having the accuracy of the sensor based method when cornering.

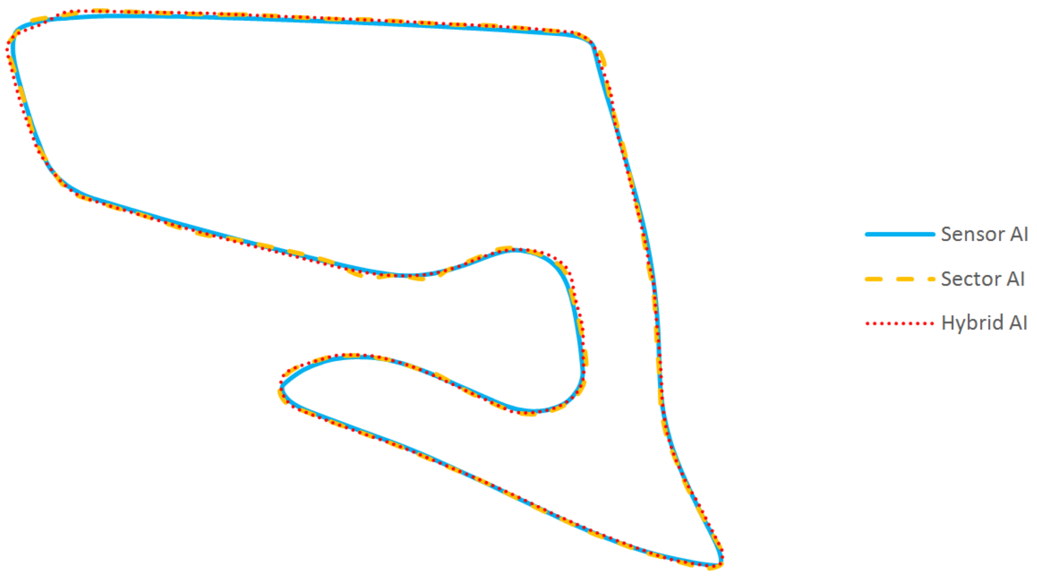


Figure 5.11: Racing lines produced by the three AI methods. The hybrid line (red, dotted) can be seen to have similar behaviours as the sector AI (yellow, dashed) on the straights, showing the ability to position itself of the edge of a track along the straights, while also exhibiting the behaviour of the sensor AI (blue, solid) in the corners by being able to hit an apex at each corner.

Unfortunately, the system doesn't perform as well in other areas. Exiting corners would benefit extra work the most on the navigation side of the system. As figure 5.11 also points out, the hybrid AI has problems with this particular part of the path finding, as a result of the waypoints being invisible from the sensor. Because of this, the AI still follows sensor method until it hits the next interface, resulting in an odd line taken from corner exits such as the first and last corners.

Collision avoidance, particularly when the AI is bunched up with opponents is an area which needs attention, as the behaviours exhibited were not believable. There is also a performance issue to focus one. This is an area for concern, as developers need to ensure that all of the target hardware is able to offer the same experience.

Another key area covered in this thesis was the set up process. For the track used, the time frame was given as three to five hours compared to the eighteen hours for the sector based method. Figure 5.12 shows the hybrid system set up for a smaller version of a bigger race track. The time taken to set this track with this system was approximately 1 hour, compared to 2 hours and 40 minutes taken by the sector method. These set ups both show that the hybrid method can take around 33% of the time taken to set the track up when compared to the sector based method. As a result of this, the tool achieved what it was designed to do, and allowed the setting up time to be reduced.

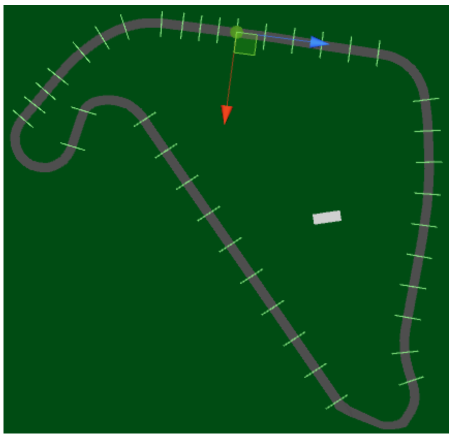


Figure 5.12: Smaller track set up with the hybrid system. This shows the smaller track ready to allow an AI that's uses the hybrid system to navigate it, with the interfaces placed on the straights, leaving the corners to be controlled by sensor steering.

The use of the set up method built for this hybrid system revealed that a dedicated process used for preparing the track for the AI can affect the time needed to complete the task. From this finding, it can be said that the set up method produce has a dramatic effect on the time required to allow the AI to traverse the track. This set up provided with this hybrid system offers a quicker option for getting AI into a racing game, as well as allowing further work to be done to it for different uses.

The point on the track in which AI brakes and accelerates isn't covered within this thesis; however, it is important to see how the AI behaves during these areas, as this can have a serious effect on the believability of the AI. Before going into the comparison, it should be noted that the braking points of these systems can be adjusted, as explained in Chapter 3.

To compare these speed profiles to highlight braking and acceleration areas, the lines used in the comparisons earlier from the first track (the original layout used in the development section of the thesis) will be used. The formula used to get these speed profiles can be seen below:

$$\frac{\sqrt{(x_n - x_{n-1})^2 + (y_n - y_{n-1})^2 + (z_n - z_{n-1})^2}}{(t_n - t_{n-1})}$$

This formula looks at the distances between two different nodes created by the AI systems in their line files. This distance change, over the time difference, will reveal the speed in which the AI would be travelling at that particular point. Figure 5.13 shows the speed profiles in the three AI systems across their laps.

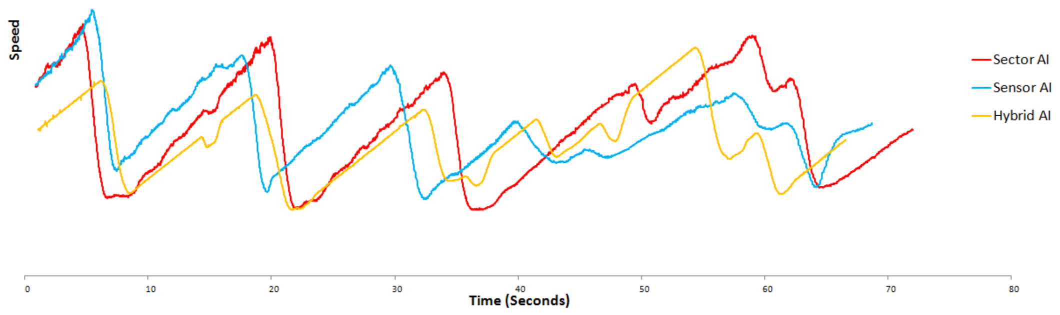
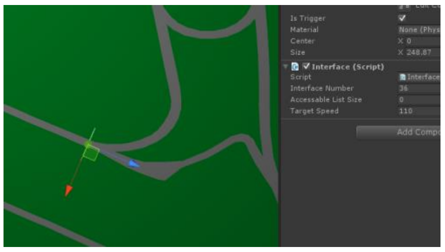


Figure 5.13: Speed profiles across the three AI systems. Notice how the line for the Hybrid method (yellow) stops before those of the Sector (red) and sensor (blue) methods, indicating a quicker lap time. As well as this, the heights of the hybrid AI line peaks lower than the previous methods, which also indicates that the AI is navigating the track at a lower speed.

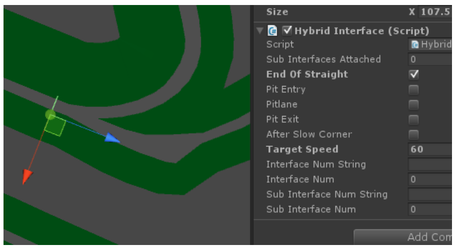
What is interesting to note is the contrast in lines by three AI systems, in particularly the hybrid system. For majority of the lap, the hybrid AI is actually slower than both systems, yet was still able to post a quicker lap time, which can be seen by its line stopping earlier. One key difference between these laps is the height variation. As the sector and sensor AI travelled along a flat racing surface, the hybrid AI had to contend with the varied height track, meaning it had to face uphill (as well as downhill) sections. The interface placement also affects this. By looking at the first braking point on this chart, we see the Sector AI braking earlier, but accelerating at roughly the same time as the hybrid AI.

By looking into the interface settings, the main culprit of this difference in speed was discovered. By looking at specific nodes on the track in both the sector and hybrid system, the target speeds were different. This is a trend that seems to be repeated across the whole track, with the straights on the hybrid system also having a slower target speed (160 compared to 190.) Figures 5.14A and 5.14B show an example of this variation.



Is Trigger	✓
Material	None (Phys
Center	X 0
Size	X 248.87
✓ Interface (Script)	
Script	Interface
Interface Number	36
Accessable List Size	0
Target Speed	110

Figure 5.14A: Interface in sector method showing target speed of 110.



Size	X 107.5
✓ Hybrid Interface (Script)	
Script	Hybrid
Sub Interfaces Attached	0
End Of Straight	✓
Pit Entry	☐
Pitlane	☐
Pit Exit	☐
After Slow Corner	☐
Target Speed	60
Interface Num String	
Interface Num	0
Sub Interface Num String	
Sub Interface Num	0

Figure 5.14B: Interface in hybrid method showing target speed of 60.

Figure 5.15 shows the speed profiles of the AI alongside that of a human player, in this case, the experienced (RG) player.

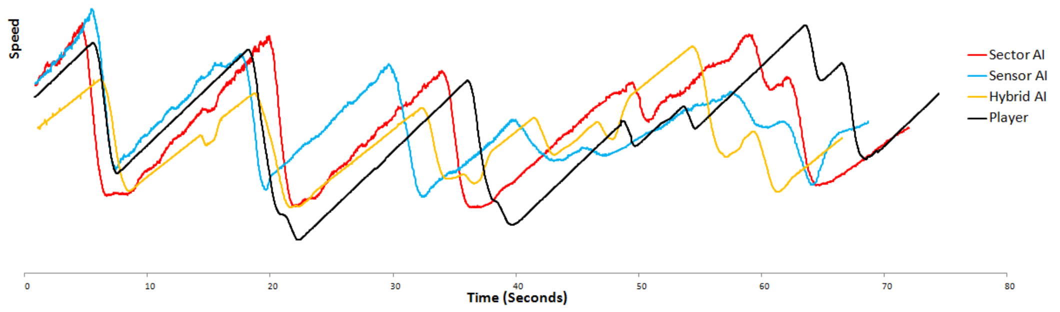


Figure 5.15: Speed profiles across the three AI systems and the player. Along with the 3 previously drawn profiles, the players speed profile (black) shows that the speed that they achieved was greater than the hybrid method (yellow) for majority of the lap, while going slower than the system in few of the corners. Their lap time was also longer than all of the AI systems.

What is interesting to take from this graph is the speed the player takes into the corners compared to the AI. For example, the player was able to take the final three corners faster than the three AI systems, with the Sector AI closely matching the player two penultimate corners. However, at the second corner, the player does double braking, resulting in a much slower entry speed.

We can see that the hybrid system is far from matching the player for majority of the lap in terms of speed. However, to ensure that the hybrid system can produce a more realistic time and speed profile, the target speeds of the interfaces would be adjusted to be similar to those of the sector AI. Figure 5.16 shows the new speed profile alongside the originals, labelled as Hybrid AI 2.

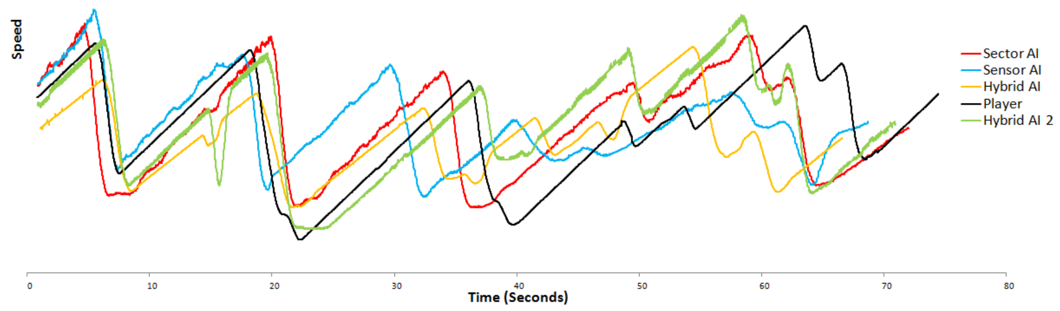


Figure 5.16: Speed profiles of the AI and player with an edited Hybrid set up. Based on the new speed profile, a few conclusions can be made. Firstly, the setup of the system can affect the speed in which the AI is able to achieve. Next, the newer version was able to hit higher speeds, which resulted in the car having to brake more for corners, increasing its lap time.

Figure 5.16 shows a much better speed profile from the hybrid system. The profile is closer to the sector and sensor based methods, and more importantly, the player. This drastic change in speed profile highlights how the interface set up conducted can heavily influence how competitive and believable the AI can be.

Chapter 6 –Conclusion

6.1 – Overall conclusion

6.1.1 – System overview

During testing, it became clear how believable the lines the hybrid system produced were compared to the other systems tested. There are key strengths and weaknesses of this system, which were highlighted throughout testing. One of the strengths is the time taken to set up a track. Out of the set up times of the three systems, the sensor based method was the quickest. The hybrid system had the second quickest time, despite having to have set up two methods, while the sector method took the longest time out of the three.

While having the ability to navigate straights like a human, the sector based method could struggle in the corners, as a result of interface and node placement. Cornering wasn’t an issue for the sensor system, as it was able to find an apex at every corner, but it does not follow a believable line on the straights like the sector method is able to. This leads into the next strength being the line believability produced, as the line followed looks more believable that the other two methods. This is a result of the hybrid system eliminating the main problems that both methods have by using each method at points where they are strongest, with the sector method used to navigate the straights, while the sensor method was in control for the corners.

Despite this, the hybrid system could be further improved to ensure that better results of a believable line are realised. One of the main areas in need of work is when switching methods from the sensor to the sector based. The primary concern for this switch over is on the exit of corners, where the AI searches for the next waypoint with the sensor. As it was realised that the waypoints are invisible to the sensors, the only way the AI can switch back is when it hits the next interface. This can bring up issues with the racing line where a drastic change in direction had to be made in the steering.

Another area which affects the believability of this system is the collision avoidance. The system is able to race well in one on one scenarios, however, it struggles when more than two cars are racing in a pack. Although part of this can be due to the performance issues with multiple AI, the logic has been set up to only deal with one opponent at a time.

Recommendations on how to improve the system are highlighted in section 6.2.

6.1.2 – Recommendations on systems use

Based on the path finding of the system, a few judgements as to when this system would be best utilized compared to other systems can be made.

6.1.2.1 –Track racing

For traditional race tracks such as the one used in demo, the AI is able to traverse multiple types of corners from hairpins to slight turns. The straights can be easily navigated with appropriate interface and waypoint placement. The set-up also suits this track type, as the total time taken is shorter than the sector based system, while still offering the same control of the line in the straights.

6.1.2.2 – Oval Track racing

Oval track racing is a discipline where competitors are racing close to/ at top speed for majority of the race. Oval track racing is unique to other forms of racing, as a racing line is rarely used by racers due to close pack racing. The racing is usually done on two lines, with a possible third line further outside if needed. These lines would be consistent for the whole track, with banked turns allowing the AI to take them at high speeds. Because of this, the hybrid system wouldn't be suited, as the sensor system wouldn't be needed to navigate the corners. The better suited system for this type of racing would be the sector based method.

6.2 – Future work

With the final section of this thesis, a few areas have been highlighted that can be worked upon.

6.2.1 – Better Physics

The hybrid system currently relies on basic physics, with the speed of the car being applied as an external force. This model had to make use of more additional scripts and objects to ensure that the car would only move in certain situations. This was despite Unity offering a physics object known as wheel colliders. These wheel colliders can perform all of this, without requiring additional objects or scripts. These wheel colliders do require set up time to ensure smooth game play. Because of this and previous experience not being positive, the decision to not use these colliders was made for this project.

6.2.2 –Line set up options

One addition to the system that could be introduced is to include multiple ways to position the waypoint nodes. Currently, the lines are created using the developed tool, with the waypoints only being altered manually. Despite this allowing the developers to get a believable line, a quicker, more effective method could be developed. One way is to allow the developers to use a player controlled car and have the waypoints be positioned as the car hits the interface.

This option not only allows for a quicker placement of the waypoints, but also produces a line taken by a player. This would be more advantageous as the line would be more representative of a human; with the option to make any small adjustments needed to the line still present. Each node can also store the speed that the player was also travelling at, assigning the value to the target speed for that particular interface, eliminating a problem discovering in the system evaluation in section 5.5.

6.2.3 – More in depth behaviours

The last part of this system that would benefit from more work are behaviours. Getting behaviours which are influenced by certain conditions would make the AI more believable, giving the user a chance of experiencing a different race each time. This would also add more to the believability of the AI with the unpredictable nature.

6.2.4 – Exploration of AI in different disciplines of racing games

One of the main focuses of this project was to explore the benefits of using a hybrid method within racing games. However, the primary focus of this project was exploring the benefits of such a system within a simulation based racing game, on the discipline of on track racing. As previously mentioned, racing games can be based on one of many different racing disciplines. One useful area that could benefit greatly into this topic is to explore the use of a hybrid method in games that offer other forms of motorsport. This would pin point the game type in which this method would be best suited, or maybe determine if the hybrid system would be a suitable option to use for a specific discipline.

References

Batchelor, J., 2014. *Race for innovation: The future of racing games | Analysis | Develop.* [Online]
Available at: http://www.develop-online.net/analysis/race-for-innovation-the-future-of-racing-games/0193671
[Accessed 10th November 2014].

Bateman, S. et al., 2011. Effects of View, Input Device, and Track Width on Video Game Driving. *Graphics Interface Conference 2011,* pp. 207-214.

Biasillo, G., 2002. 9.1 Representing a Racetrack for the AI. In: S. Rabin, ed. *AI Game Programming Wisdom.* Hingham, Massachusetts: Charles River Media, pp. 439-443.

Butz, M. V., Linhardt, M. J. & Lönneker, T. D., 2011. Effective Racing on Partially Observable Tracks:. *IEEE TRANSACTIONS ON COMPUTATIONAL INTELLIGENCE AND AI IN GAMES,* 3(1), pp. 31-42.

Codemasters. 2014. *F1 2014*. [DISC] PlayStation 3. Southam, Warwickshire, Codemasters Software Company Limited.

Codemasters. 2015. *F1 2015 - Offical Game Features Page.* [Online] Available at: http://www.formula1-game.com/us/features [Accessed 24 March 2016].

Conroy, D. & Wyeth, P., 2011. Building better bad guys: a new framework for game AI design. *Proceedings of the 7th Australasian Conference on Interactive Entertainment,* pp. 1-3.

Develop, n.d. *Online racer Quantum Rush soon as a single player version? | Games industry press releases | Develop.* [Online]
Available at: http://www.develop-online.net/press-releases/online-racer-quantum-rush-soon-as-a-single-player-version/0192635
[Accessed 28 August 2015].

Driving Experiences. 2012. *Racing Line* |. [Online] Available at: http://www.driving-experiences.co.uk/racing-line/ [Accessed 4 December 2015].

Formula1-Dictionary.net, n.d. *Downforce.* [Online] Available at: http://www.formula1-dictionary.net/downforce.html [Accessed 14 October 2015].

Fujii, S., Nakashima, T. & Ishibuchi, H., 2008. A Study on Constructing Fuzzy Systems for High-Level Decision Making in a Car Racing Game. *2008 IEEE International Conference on Fuzzy Systems*, pp. 2299-2306.

Guo, R. & Quarles, J., 2012. Exercise-Based Interaction Techniques for a Virtual Reality Car Racing Game. *IEEE Virtual Reality 2012,* pp. 93-94.

Land, M. F., Tatler, B.W., 2001. Steering with the head: The visual strategy of a racing driver. *Current Biology*, Volume 11 (15), pp. 1215- 1220.

LemonTubeAmiga. 2012. *Formula One Grand Prix (F1GP) (Amiga) - A Track Guide and Review - by LemonAmiga.com* [Video] [Accessed 27 September 2015] Available from: https://www.youtube.com/watch?v=LY39_a1DMMA

Loiacono, D., 2012. Learning, Evolution and Adaptation in Racing Games. *CF '12 Proceedings of the 9th conference on Computing Frontiers,* pp. 277-284 .

Loiacono, D., Cardamone, L. & Lanzi, P. L., 2011. Automatic Track Generation for High-End Racing. *IEEE TRANSACTIONS ON COMPUTATIONAL INTELLIGENCE AND AI IN GAMES,* 3(3), pp. 245-259.

Makuch, E., 2014. *PS4-exclusive Driveclub did experiment with virtual reality, but that's not why it was delayed - GameSpot.* [Online] Available at: http://www.gamespot.com/articles/ps4-exclusive-driveclub-did-experiment-with-virtual-reality-but-that-s-not-why-it-was-delayed/1100-6418417/ [Accessed 15 October 2015].

Martin, G., 2014. *Titanfall Review (Xbox One / 360 / PC) :: Games :: Reviews :: Paste.* [Online]

Available at: http://www.pastemagazine.com/articles/2014/03/titanfall-review-xbox-one-pc.html
[Accessed 11 October 2015].

Microsoft, 2015. *Drivatar™ in Forza Motorsport - Microsoft Research.* [Online] Available at: http://research.microsoft.com/en-us/projects/drivatar/forza.aspx [Accessed 03 October 2015].

Most Reliable Car Brands. 2016. *Grand Prix of Australia to 2016: Results of the Race | Most Reliable Car Brands*. [Online] Available at: http://www.mostreliablecarbrands.com/grand-prix-of-australia-to-2016-results-of-the-race/ [Accessed 5 April 2016].

Museum of the Game, n.d. *Pole Position - Videogame by Atari.* [Online] Available at: http://www.arcade-museum.com/game_detail.php?game_id=9063 [Accessed 25 September 2015].

Northern Motorsport, 2012. *Glossary of Racing Terms.* [Online] Available at: http://www.motorsport.in/glossary-of-racing-terms/ [Accessed 22 August 2015].

Paradigm Shift Driver Development, 2015. *Racing and motorsports Terms Glossary - Paradigm Shift Driver Development.* [Online] Available at: http://www.paradigmshiftracing.com/setup-guide/racing-and-motorsports-terms-glossary
[Accessed 14 October 2015].

PlayStation Mobile Inc., 2015. *Official PlayStation Website | PlayStation.* [Online]
Available at: https://www.playstation.com/en-gb/
[Accessed 1 November 2015].

Polyphony Digital. 2013. *Gran Turismo 6*. [DISC] PlayStation 3. Tokyo Japan, Sony Computer Entertainment.

Rouse, R., 2005. Chapter 9: Artificial Intelligence. In: *Game Design Theory & Practise Second Edition.* Plano, Texas: Worldware Publishing, pp. 151-171.

Rouse, R., 2005. Glossary. In: *Game Design Theory & Practice Second Edition.* Plano, Texas: Worldware Publishing, pp. 655-671.

Rudderham, T., 2015. *Project CARS 2.0 patch goes live with Oculus Rift improvements.* [Online]
Available at: http://www.theriftarcade.com/project-cars-2-0-patch-goes-live-with-oculus-rift-improvements/
[Accessed 15 October 2015].

Ryan, P., 2015. *Quantum Rush Champions Review Screenshot 1 | Brash Games.* [Online]
Available at: http://www.brashgames.co.uk/2015/09/05/quantum-rush-champions-review/quantum-rush-champions-review-screenshot-1/
[Accessed 19 September 2015].

Schwab, B., 2009. Chapter 11 Racing Games. In: *AI Game Engine Programming.* Boston: Course Technology, pp. 191-201.

Scullion, C., 2016. *Best F1 games: The top Formula 1 titles of all time* . [Online] Available at: http://www.redbull.com/uk/en/games/stories/1331653281568/best-f1-games-formula-1-f1-2013 [Accessed 24 March 2016].

Slightly Mad Studios 2015. *Carrer – Project CARS.* [Online] Available at: http://www.projectcarsgame.com/career.html [Accessed 24 March 2016].

Slightly Mad Studios 2015. *Locations – Project CARS.* [Online] Available at: http://www.projectcarsgame.com/locations.html
[Accessed 23 March 2016].

Slightly Mad Studios. 2015. *Project CARS.* [DISC] PlayStation 4. London, Slightly Mad Studios.

Smith, A., 2014. *Under-steer vs Over-steer | Journey to the Drift World.* [Online] Available at: http://wake-turbulence.blogspot.co.uk/2014/10/under-steer-vs-over-steer.html
[Accessed 14 October 2015].

Snow, B., 2011. *Why most people don't finish video games.* [Online] Available at: http://edition.cnn.com/2011/TECH/gaming.gadgets/08/17/finishing.videogames.snow/
[Accessed 28 August 2015].

Tang, H., Tan, C. H., Tan, K. C. & Tay, A., 2009. Neural Network versus Behavior Based Approach in Simulated Car. *Evolving and Self-Developing Intelligent Systems, 2009.*, pp. 58-65.

Treyarch. 2010. *Call of Duty: Black Ops.* [DISC] PlayStation 3. Santa Monica, California, Activision.

Unity. 2013. DeltaTime – Unity Official Tutorials [Video] [Accessed 10th October 2015] Available from: https://www.youtube.com/watch?t=1&v=a-w7w8x_moE

Unity, 2015. *Unity - Manual: Prefabs*. [Online] Available at: http://docs.unity3d.com/Manual/Prefabs.html [Accessed 4 December 2015]

Wang, J.-Y. & Lin, Y.-B., 2010. An Effective Method of Pathfinding in a Car Racing Game. *Computer and Automation Engineering (ICCAE), 2010 The 2nd International Conference*, Volume 3, pp. 544-547.

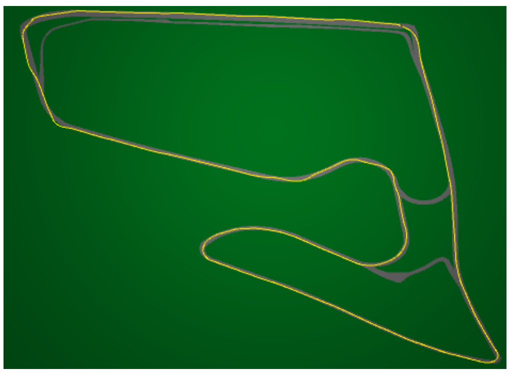
Wang, J.-Y. & Lin, Y.-B., 2012. Game AI: Simulating Car Racing Game by Applying Pathfinding Algorithms. *International Journal of Machine Learning and Computing*, Volume 2, pp. 13-18.

Appendices

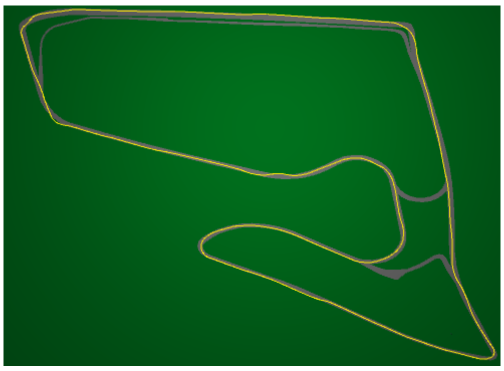
A – Variation in waypoint navigation

Various lines produced by the waypoint based system, showing inconsistencies with the navigation of the NPC. This Appendix shows of a variation of lines produced by this navigation system, why obvious different can be seen at certain points of the track.

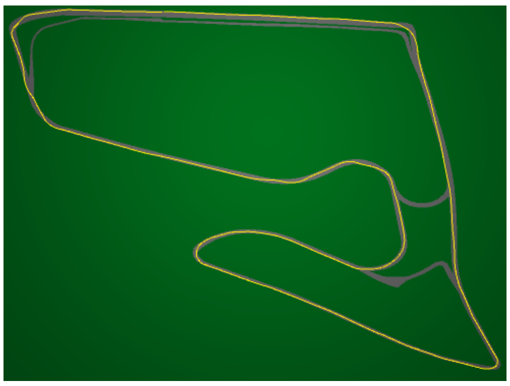
A.1 – Example Line 1



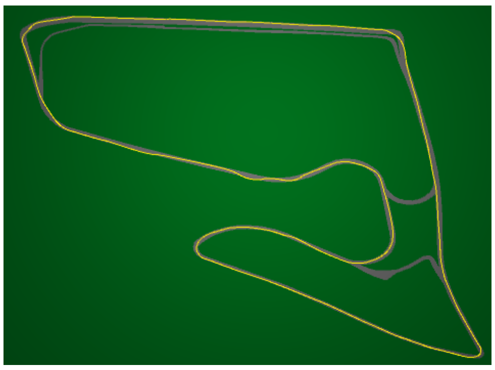
A.2 – Example Line 2



A.3 – Example Line 3



A.4 – Example Line 4

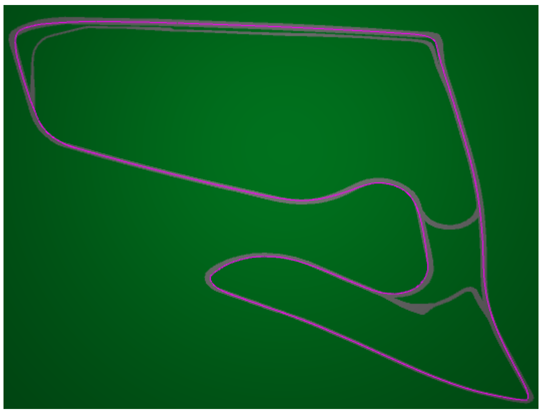


Although not the perfect example of a racing line produced by this system, this was the line that was used for the comparison of lines between humans and AI systems.

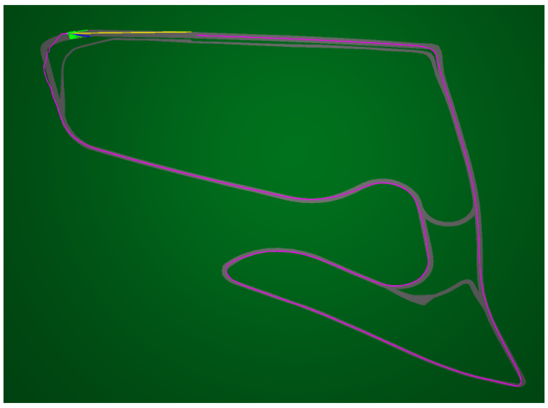
B – Sensor based system: Fast vs. Slow Hardware

The differences in performance of the sensor system shows that on slower hardware, the sensor method had difficulties recording a line on file, with B.2 taking an interesting line at the last corner.

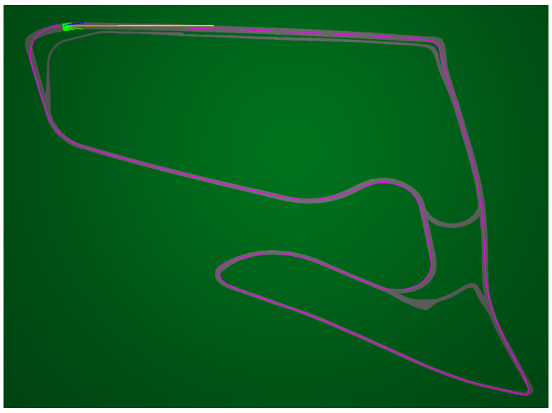
B.1 – Line drawn on faster hardware



B.2 – Line one drawn on slower hardware



B.3 – Line two drawn on slower hardware



C – CD