Applying neuroevolutionary algorithms to video game artificial intelligence agents

UNIVERSITY OF
LINCOLN

Ashley Knowles

KNO15559083

School of Computer Science
University of Lincoln

A dissertation submitted in partial fulfilment of the requirements for the degree of
BSc (Hons) Games Computing

2018

Abstract

A long standing challenge within game development is the ability to create immersive environments with artificial intelligence (AI) agents that are able to maintain a convincing illusion of consciousness with minimal computational overhead. Traditional AI methods rely on simplification and prediction to achieve this, however these often lack in path diversity and have difficulty producing behaviour which a human player would interpret as intelligent. This paper investigates the effectiveness of artificial neural networks trained using the NEAT evolutionary algorithm which mimics the behavioural systems found in biological nature, to produce a realistic façade requiring little processing power. A group of participants were asked to rate their experience playing against an unknown AI opponent including questions about whether they thought they were playing against a human or AI. Results appeared to show conflicting evidence that the method applied in such an application was able to produce a convincing illusion of intelligence, as the quantitative tests demonstrated that the method was effective while inversely the qualitative tests indicated that participants felt there were too many problems with the system.

Contents

List of Figures

Chapter 1

Background and Literature Review

1.1 Introduction

1.1.1 Keyword definitions

- **Artificial Intelligence (AI)**
  The simulation of human intelligence processes by a computer such as perception and decision making operations

- **Artificial Neural Network (ANN)**
  A computing system comprised of a group of connected nodes similar to neurons found in the human brain

- **Evolutionary Algorithm (EA)**
  A set of search and optimization heuristics used to evolve a NN

- **Machine Learning (ML)**
  The field of computer science that deals with computer systems having the ability to "learn"

1.1.2 Background

It is often the case within video game development that the need arises for an AI system that can accompany an immersive experience. In these cases, the system must be capable of producing behaviour that has the potential of fooling a human player into believing that it is in fact interacting with another human. Designing such a system in most cases first requires knowledge of human behaviour so that it may be reproduced, however the behaviours are often reduced and simplified to save processing time and as a result are not quite able to convince humans. This paper will investigate the application of machine learning combined with evolutionary algorithms within a simple video game test bed in producing an opponent which demonstrates believable characteristics that can convince test participants that they are playing against a human.

The work produced by this investigation could have potential applicability in genres which require AI to exhibit behaviour similar of that of a human. An example of this is in games such as Spy Party or Assassin's Creed where a player is surrounded by a crowd of AI and has to attempt to blend in, while another guesses which is the human. In these cases, it's essential for the AI to not give away any tell-tale signs that it is a computer controlled agent.

1.2 AI Believability

It can often be hard to determine how effective an artificial intelligence agent is in producing a convincing and believable illusion of conscious decision making. Many prior studies have attempted to solve this problem by producing a method for quantifying believability. Livingstone (2006)[1] suggested that the traditional Turing test that is often used to determine whether a human can tell the difference between an AI agent or a human player is no longer sufficient in video game or research contexts, as the constraints of the test do not provide a detailed insight into which areas the AI performed well. The paper goes on to explain that while the goal of researchers is to understand and explore intelligent behaviour to create something of "substance", often the contrary goal for video game designers is to create a realistic "facade" which is capable of tricking players. As a result of this, in order to generalize the imitation game created by Turing and remove some of its restrictions, a set of believability criteria were proposed and expanded on from the findings of Laird and Duchi (2000)[2] and Wetzel (2004)[3] who attempted to find the most common failures observed in AI. While the paper suggests that the Turing test is no longer relevant in a research context, it could however be used to attain a simple metric from experimental participants who are not familiar with the complexities involved with AI. The proposed alternative criteria has not yet been used in neural network based research before, as is the case with the investigation that will be conducted. By using this criteria within the participant questionnaires, it will be possible to attain valuable metrics which can be later analyzed to conclude the significance of the various different sections that make up the overall AI system, as well as any potential shortcomings that may become apparent within the artifact.

Tence, Buche, De Loor and Marc (1992)[4] studied the problem of producing believable virtual characters within a video game context. They defined and considered believability as the agent "giving the feeling of being controlled by a player". The paper finds that the best way to meet the criteria of convincing players is to use a technique such as reinforcement or imitation learning, but goes on to explain that imitation learning can often be difficult to achieve

as it's easy for the agent to learn to copy the actions of its teacher exactly rather than adapting dynamically to its environment. It will be important to take this information into consideration when designing the experiment for the proceeding investigation. While the idea of imitation learning does sound enticing for producing results which mimic that of a human, it's possible that the AI will learn the specifics of the track and copy the way that the human plays the game. For this reason, it is likely that this technique will not be used, but instead a more generalized approach will be taken that can be applied to various different gameplay scenarios.

1.3 Machine Learning and Evolutionary Techniques

Several prior studies have found methods for integrating various forms of machine learning techniques into a video game context. Stanley and Miikkulainen (2002)[5] established the NeuroEvolution of Augmenting Topologies (NEAT) framework to allow agents to solve complex reinforcement learning tasks in a way that simulates real world genetics including species and generations. This paper is widely regarded as the de facto standard of the neuroevolution method, and as such the investigation would benefit from the use of a basic implementation of this.

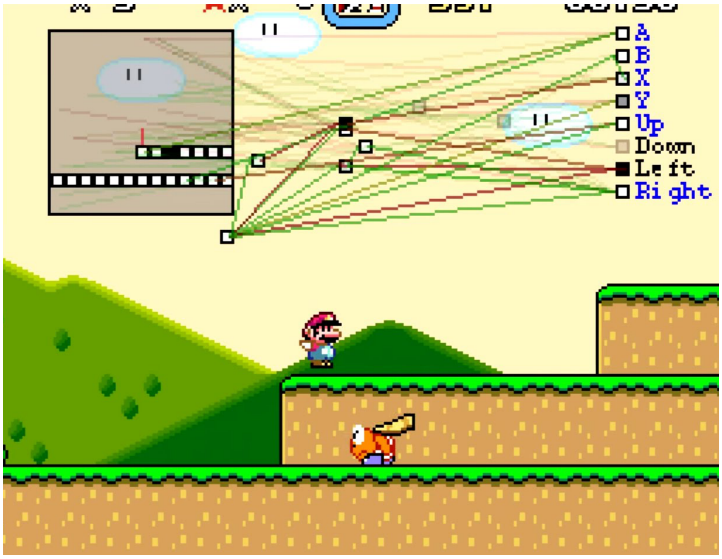


Figure 1.1 A screenshot of MarI/O[6], an implementation of NEAT in Super Mario Bros

The figure above shows a screenshot of MarI/O[6], an experiment created to see if it was possible to get a character to complete a stage of the Super Mario World game, using only the NEAT technique. Each iteration of the evolutionary learning process tries different combinations of outputs until the fitness function increases. From this, a network, similar to the

neurons of a brain, begin to build structures that link each node together to create a complex network that gives a semblance of intelligence. As it's been shown that this implementation has potential within a video game setting, this experiment will attempt to replicate, albeit with adjustments to suit the specific requirements of the project, the results of this novel method.

Faced with the challenge of producing AI capable of playing various different video games with little to no domain-specific knowledge, Hausknecht, Lehman, Miikkulainen and Stone (2013)[7] applied four different neuro-evolution algorithms to 61 different Atari games. The final results demonstrated that NEAT (and similarly the subset HyperNEAT) outperformed both Conventional Neuro-evolution (CNE) and Covariance Matrix Adaptation Evolution Strategy (CMA-ES) in object and noise models at attaining the highest fitness value, which determines how well the AI has progressed into the game. The main difference between NEAT and the aforementioned algorithms is the addition of feed-forward architecture, where the network is able to dynamically adjust the weightings of the nodes by passing information from the input layer to the hidden layer nodes. The study also found that in most cases human experts were able to significantly outperform planning and random algorithms which were used as a base, where the neuro-evolution algorithms were able to achieve higher than average scores in multiple games. While the models were compared to human scores, they were assessed independently rather than simultaneously and this is an area to be further explored.

While adaptive AI has seen numerous implementations, Ponsen and Spronck (2014)[8] validated their proposal that offline evolutionary algorithms can be used to enhance the performance of traditional adaptive game AI, by extending the algorithm's domain knowledge dynamically. To test their proposal, they ran two different versions of AI, the first being Dynamic Scripting and the second the offline evolutionary algorithm. Being able to expand on the original rule base that was manually defined, the enhanced AI showed significantly improved performance when applied in a Real-Time Strategy Game (RTS) environment.

Chapter 2

Methodology

2.1 Project Management

The proper management of the overall scope of the project was critical to ensuring that the work was completed on time and to a satisfactory standard. There are several standard approaches to methodologically achieving this, each with their own merits and disadvantages. The specific requirements of the project were such that a game artifact needed to be produced to a degree where it was capable of being played, tested and analysed within a relatively short period of time. Alongside this, some elements of the game design needed to be constantly deloped and tested to see their effectiveness, and for this reason it was decided that the most appropriate methodology would be to use an agile iterative approach.

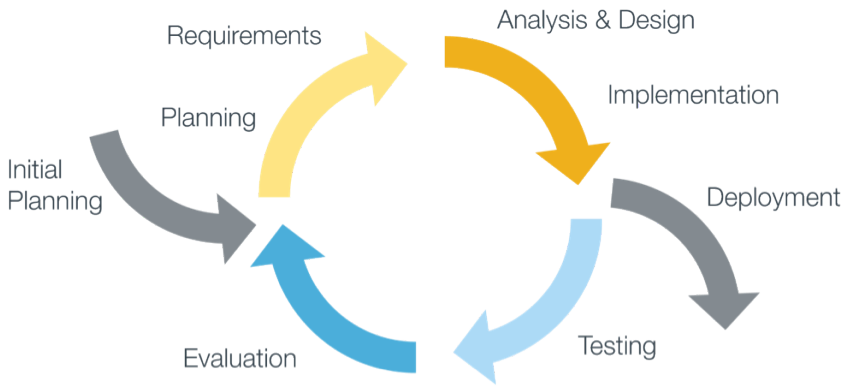


Figure 2.1 Iterative agile development cycle

Above is a figure detailing the various processes contained within the methodology which were adhered to for the most part, to keep the project on track.

- Planning / Requirements
  The planning and requirements phase involved taking the initial concept and expanding it to form a coherent idea which had the potential to be developed for the final game artifact. Before the design phase, it was necessary to research existing technologies and machine learning techniques to see which could be applied or developed on during the implementation phase. Finally, the planning phase allowed for time allocation in the

form of a Gantt Chart, laying out blocks of time for each section of the project to ensure it stayed on time.

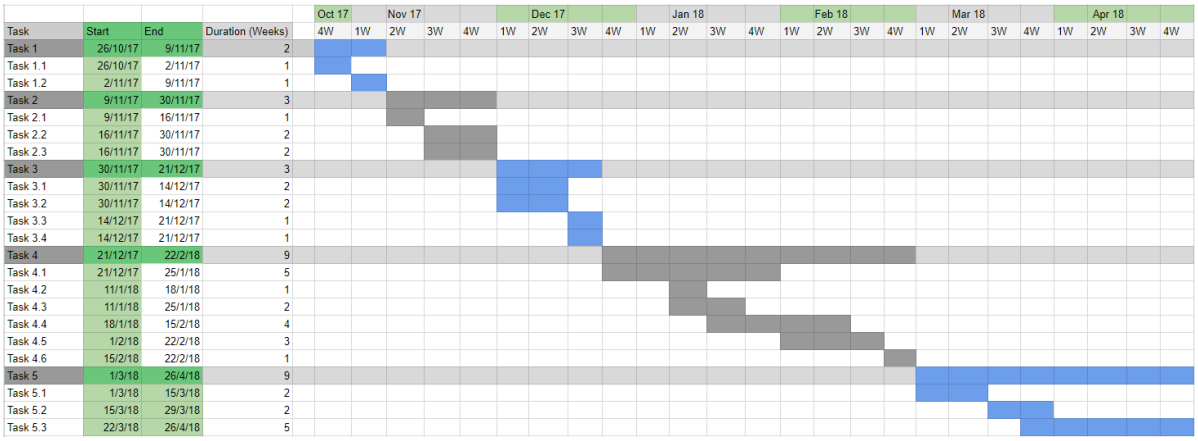


Figure 2.2 Gantt Chart time allocation

- Design
  The design phase of the project dealt with the main parts of the game that required any significant amount of design work such as deciding on a method for evolving a network as well as the overall track design for the race. To begin with, none of the implementation had started, however due to the iterative nature of the chosen methodology, it was possible to come back to this at a later stage and test out new parts of the system.

- Implementation
  This stage of the methodology dealt with substantiating the original design and implementing it within the chosen game engine. This meant getting any algorithms or libraries to work within the actual game.

- Testing Testing the game involved making sure that each element of the game worked correctly, including the mechanics as well as the frames per second to ensure that it ran well on all machines that participants would play it on.

- Evaluation
  Finally, the evaluation entailed a self-reflection about what went well in the project, what went wrong, and what would have been differently in the future given less constraints.

These 4 stages comprised the majority of the investigation and ensured that time was spent in the proper areas.

2.2 Software Development

There were many specific demands that needed to be taken into consideration when determining which software development methodology to use for the project. The nature of the project is such that an iterative process would be beneficial, allowing for changes to be made to the design at any stage in the development process.

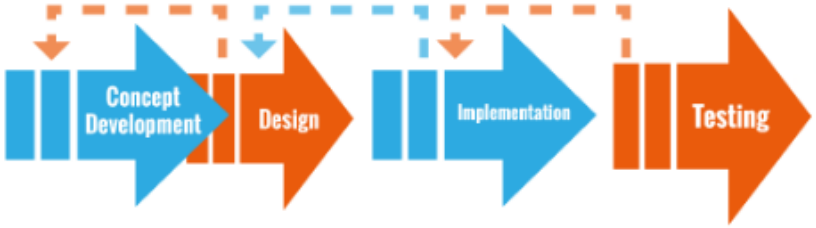


Figure 2.3 Iterative agile game development cycle

The figure above shows a standard iterative design pattern which shares many similarities with the system in the previous section, in that the design, implementation and testing headings are still present. In this context however, it focuses primarily on the software being designed rather than the project as a whole.

As neural networks are unpredictable and experimental by design, tweaking and testing different variables is essential to ensure proper operation that can be tailored specifically to the game's needs, and as such an iterative design process would lend itself well to the project in this regard.

Another aspect to be considered is that the development process is being undertaken alone, rather than with a group. If this weren't the case however, a methodology such as SCRUM would be used, converting each potential feature into a user story to be later sorted, prioritized and implemented in an agile fashion.

As previously mentioned, implementing a neural network within a video game is a chal-

lenging process that changes with each project and implementation, with no single defined structure. A waterfall structure was considered for the implementation, and while it would be useful for creating the general racing game, the requirements and limitations of the AI were unknown, and required a more iterative process.

For these reasons, an agile iterative process was chosen as the software development methodology. Poppendieck (2001) described a list of 10 practices which make Lean Manufacturing successful and explains how they can be used in a software development context. Several of these practices were applied to the project, by eliminating or optimizing consumables such as diagrams and models that do not add value to the final deliverable, and the use of iterative development to reduce development time as the project is time sensitive, requiring a lot of time for data collection and analysis rather than development.

2.3 Toolsets and Machine Environments

There are a wide variety of different game engines and tools that had their advantages and disadvantages for the development of the final artifact. Key points that needed to be taken into consideration were as follows:

- External library support
  External library support was essential to completing the artifact. Neural networks are built using several lengthy and complex algorithms, and due to the time constraints placed within the project, it was decided to use an external library which contains some of the required functionality. Doing this allowed for time to be spent tweaking and testing values rather than creating the entire code base from scratch.

- 3D engine capabilities
  During the initial planning and requirements phase, it was determined that a 3D representation of the race course would be more beneficial and intuitive to the players rather than playing in 2D, and for this reason an engine with 3D graphics capabilities was required.

- Multiple device support flexibility
  In order to get as many participants as possible to test the artifact, having the flexibility able to run the game on different systems and hardware would be useful.

- Fast prototyping
  Perhaps the most important feature requirement for the game engine was the ability to rapidly prototype different versions of the artifact. It was incredibly useful to produce multiple versions with slight changes in code to test different features of the artifact.

Game Engine	Library support	3D Engine	Multiple devices	Fast prototyping
Unity	Yes	Yes	Yes	Yes
Unreal Engine	Yes	Yes	Yes	No
Urho3D	No	Yes	No	No
CryEngine	Yes	Yes	No	No
Monogame	Yes	Yes	Yes	No
OpenGL	Yes	Yes	Yes	No

A study conducted on the unique challenges of game development Koepke et al (2013)[9] describes how there are various constraints in place such as the subjective system requirements, complex interactions as well as testing challenges. The paper goes on to give a "review" of popular game engines including CryEngine and Unity3D, which influenced the final decision about which was to be used in creating the game.

Several game engines were tested before developing the artifact, with the aforementioned requirements being taken into consideration. Above is a matrix table demonstrating the applicability for each engine with regards the project. While the visual aspect of Unreal Engine is a strong point, importing libraries is a difficult task, and prototyping takes time. In the case of CryEngine, producing a commercial product requires a full version of the engine, and royalties paid toward the developers which was not possible given the budget constraints for the investigation. Urho3D, Monogame and OpenGL all required each visual aspect to be hard coded, with no visual editor included as standard, and this was a requirement for fast debugging and prototyping.

The decision was eventually made to use the Unity engine, as it allows for fast prototype creation as well as supporting multiple different neural network libraries.

After the Unity engine had been selected, finding a neuroevolution library that had cross-compatibility with the engine was the next task. The Unity engine supports multiple scripting languages such as CSharp and Javascript, and thus finding a suitable library which contains all of the required functionality was not a difficult task. After researching, it was found that the most up-to-date version of the NEAT implementation was SharpNEAT[10], which was

originally designed as a standalone library for neuroevolution, however it was ported to the unity engine in a version known as UnityNEAT. After testing, it seemed to provide all the required functionality and was selected as the library of choice.

2.4 Research Methods

The hypothesis for the experiment is that, under controlled conditions, participants will not be able to tell if the opponent that they play against is a human or computer controlled. To test this hypothesis, a study will be designed which attempts to exhaust all the necessary means of data collection, including including the believability criteria defined by Livingstone (2006)[1] as originally mentioned in the literature review for this investigation. The criteria proposes a set of questions to ask participants which give a quantifiable score to each section of the AI being reviewed with regards to believability.

For this experiment, both qualitative and quantitative data sets are required to gain and accurate representation for how the participant will feel about the opponent that they play against. For qualitative data, the believability criteria will be condensed to only the parts that are relevant to this study and posed in the form of a likert-scale, which provides a good base for statistical analysis such as spread. On the other hand, qualitative open-ended questions will also be used to allow the participants to give their own specific feedback about what about the opponent they thought worked well and what didn't.

The data obtained from the likert-scale questions will be subjective and ordinal, thus requiring specific means of statistical analysis. The results will be represented in the form of a bar chart, as it shows the amount of responses for each value, on each question.

The dependent variable for the experiment is how likely the participants are to be fooled by the experiment, and the independent variable is the type of AI that will be used.

Chapter 3

Design, Development and Evaluation

3.0.1 Requirements and Design

In order to evaluate how believable the method of producing an evolutionary trained AI agent is, a game was created which features an opponent car which players would race against around a course for several laps before completing a questionnaire rating their experience, including how convinced they were that the enemy car was an AI or human player. The first step in designing the game was decide on a neural network architecture which was both easy to implement and suited the specific needs of the game.

It was important to first understand the underlying nature of neural networks and how they operate to select a fitting model. A basic network consists of multiple input nodes as well as several hidden nodes and output nodes. The hidden nodes (sometimes also referred to as neurons), are grouped into layers and interact with each other through connections that have a weighting value associated with them.

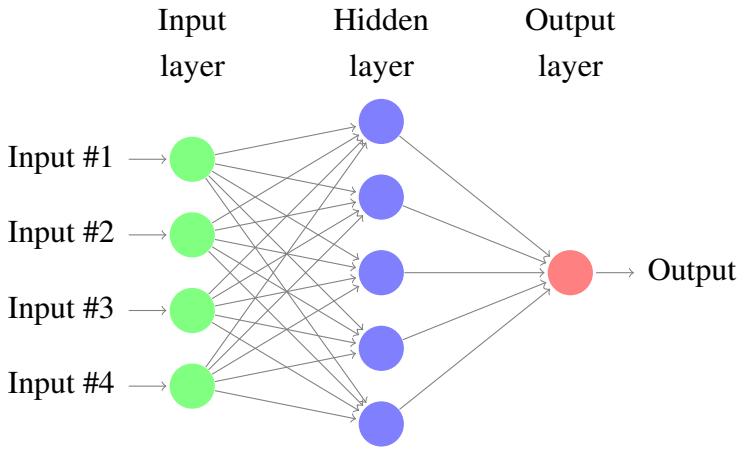


Fig 1.1: An example of a traditional neural network with 4 input nodes, a hidden layer containing 5 nodes and a single output node.

Values from the input layers are passed to the nodes within the next layer (hidden layer) after being multiplied by the weighting of the connected node. For each node in the next layer, the sum of each connection is calculated with the addition of a "bias" value specific to that

node. Finally, the node's value is passed through an activation function to transform the value between 0 and 1 before it is passed to the next layer.

In traditional neural network topologies such as a Multi Layer Perceptron (MLP) or Recurrent Neural Network (RNN), the architecture and weighting of the nodes that connect the input layers to the output layers are either decided by hand or given random values, creating a fixed topology that never changes throughout its lifetime. Methods such as back-propagation or other forms of stochastic gradient descent have been used to adjust the weightings of each node dynamically and allow the network to "learn" to correct its output.

Stanley and Miikkulainen (2002)[5] however, suggest a novel way of evolving a neural network in a population-based manner, mimicking traits and mechanisms found in biological evolution such as species selection and mutation. An initial population is created with few to no neuron connections, and gradually builds complexity over time as the network tries seemingly random output activations until a fitness function increases. As the fitness function increases and the agent progresses further into the game, the genome for the population is stored and a new, slightly mutated population is created until a more complex network is evolved. While traditionally this technique has been used to simulate biological organism evolution, many of the same principles can be applied to AI simulation applications, including race cars as was done in this study.

3.0.2 Implementation

Several requirements were considered when implementing the NEAT method into the game. The first consideration was about the inputs that should be passed to the network. The car must be able to get a clear image of its surroundings in order to respond appropriately to the external stimulus. To achieve this, an array of "feeler" sensors were attached in 5 regions of the car which enable it to see oncoming collisions as well as their distances.
As the car will likely only be able to drive in the forward direction, to simplify the network as much as possible sensors were placed only on the front, corners and sides. The sensory information from these served as the input nodes which were later processed by the hidden nodes.

The outputs from the network were mapped to the basic controls of the car such as braking, accelerating and steering. To again simplify the network, a single output was used to steer the car, with a value above 0.5 being left and below being right. A similar setup was also be used for accelerating and braking.

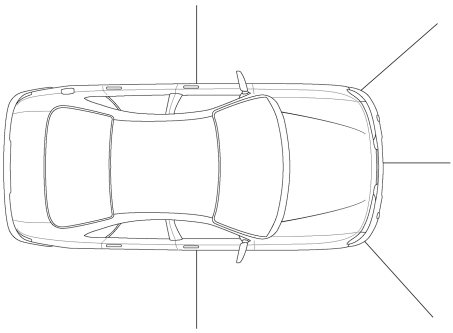


Figure 3.1 An illustration showing the locations of the car sensors

The final consideration for the system was the fitness function. This was a combination of factors that were used as a metric for how well the AI was performing around the track that allowed the network to adjust its weighting and evolve a newer, superior generation of car. The formula unified several metrics such as:

- Current lap
  The number of laps that the car has travelled.

- Current track piece
  The pieces of track on the circuit will be numbered, and the current track piece that the car is on will be considered.

- Number of wall hits
  Should the car hit a wall, points will be deducted.

The training process for learning using an evolutionary algorithm can often take long periods of time as new generations are created, known as epochs, with mutations which lead to an increase in agent performance. A method to speed up this process is to create multiple separate instances of the car, spawned at the same time, with an increased timescale to lower the training time to a few minutes rather than hours.

When creating the course layout for the race, there were multiple design routes that could have been taken. A procedural race course was initially considered, as this would allow a new experience to be created for each participant, however given that each participant would only be playing the game for the duration of a single lap, it was an unnecessary complication in the

design process and as such it was decided that a single static level was more appropriate, which also aided in unifying the experience between each player to give more consistent and reliable test results.

After it was decided that a single level was to be created, the specific requirements for the research that was being carried out and the data that was to be captured had to be taken into consideration in the track layout. The course needed to be able to accommodate a wide range of tests which would be able to push the AI to its limits as well as giving the player an accurate representation of the nature of its abilities. An article by McMillan (2011), proposed a set of 5 metrics for course design which claim to give ordinary players who might not have any experience in race car driving, an arcade-like and accessible experience. Basing the design of the course on this set of guidelines seemed to be an appropriate solution to the task at hand. The metrics, as well as the specific interpretation for each of them that were used in the experiment are as follows:

- Metric 1: Race Line
  The race line is the most optimal path that a driver can take through the course which results in the fastest lap time. The figure above shows the skeleton of the entire race

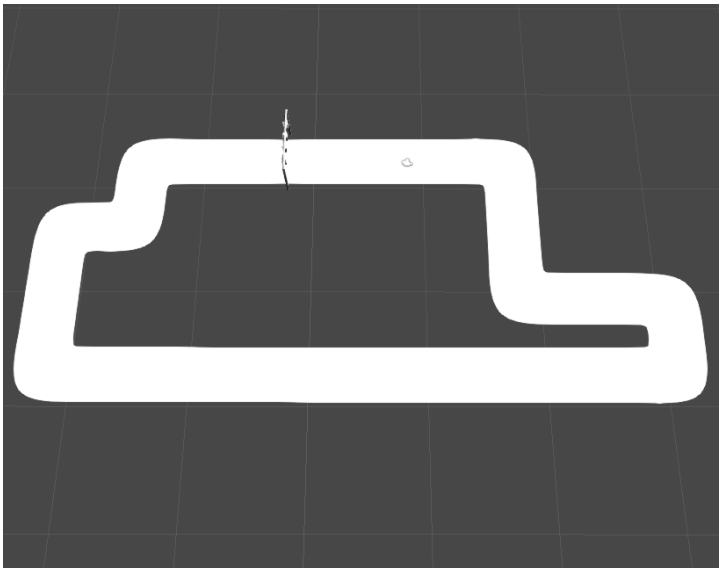
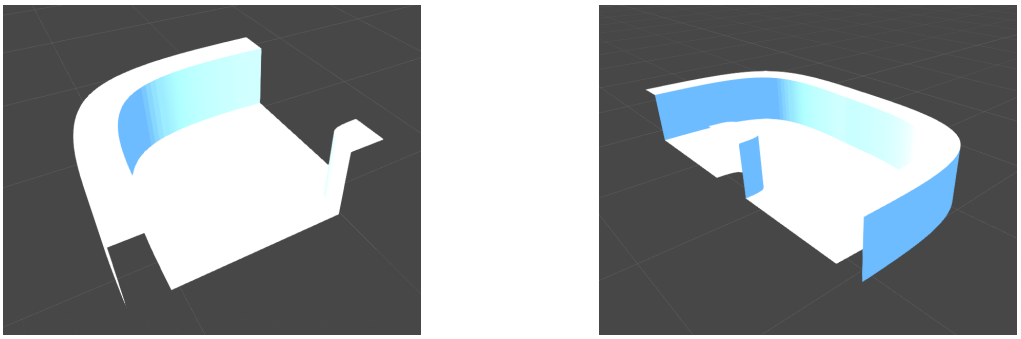


Figure 3.2 The entirety of the track

  circuit that was used in the experiment. It's clear that at each corner of the track there is an optimal path to be taken which takes the driver through the track the fastest.

- Metric 2: Clipping points (and related metrics)
  A clipping point is a target that drivers should aim for in the middle of a turn which places the least amount of lateral (sideways) force on the car whilst allowing for the shortest possible route. With a lengthy amount of training time, the AI should eventually learn to aim for this point, giving it the ability to take a shorter route than the human participant, and perform overtaking maneuvers. To save time during the development process, a single bend was created which could be copied and rotated to allow for faster map creation, without spending time manually 3D modelling each turn. The model that was created served as a general purpose bend which gave the AI the ability to take both long and fast routes and demonstrate its decision making capabilities.



(a) Corner 3D model (b) Joining 2 corners

Figure 3.3 A corner piece by itself, as well as being rotated 90 degrees and fitted together

  The figures above show how a curved section of track can be rotated in 90 degree increments and fit together with another piece to allow a turn within the track. The wall pieces that were added to the sides serve as triggers for the AI sensors, which tell it when a collision is about to occur. To optimize and increase game performance, these were eventually hidden, however the collision boxes remained.

- Metric 3: Track Width
  The track should be large enough in width to allow for overtaking opportunities, and in general, the wider the track, the easier it is to drive around as it creates a deeper clipping point angle as well as keeping drivers away from the walls. It was eventually decided to use a static width for the entire track to both save time modelling as well as making the user experience as easy as possible to allow the for most people to participate regardless of experience level.

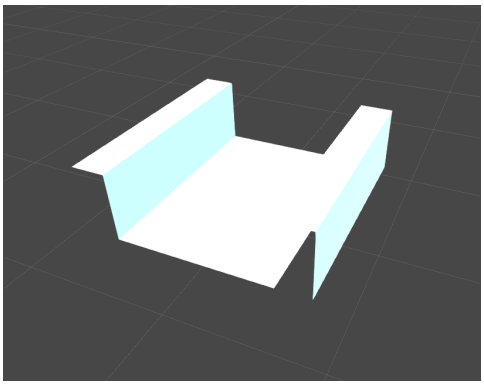


Figure 3.4 A section of track demonstrating track width

- Metric 4: Camber
  The camber metric refers to the car's ability to under-steer or over-steer in a during a corner, based on the angle of attack at which the car enters the turn. As previously mentioned, a single corner was created that was used for the entirety of the track. In this case, as the car physics did not allow for drifting or sliding to make the experience more user friendly and lower the difficulty curve, it wasn't necessary to take this metric into consideration.

- Metric 5: Height Variation
  Height variation simply refers to changes in elevation throughout the track. Again, to simplify the experience, no height changes were made during the track as inexperienced players could have trouble navigating the course.

To allow the opponent to demonstrate its ability to make decisions based on its surroundings, a small obstacle was modelled which splits the track into 2 pieces and forces the car to follow one of the paths.

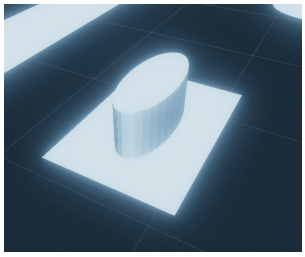


Figure 3.5 The obstacle placed within the track

With the track design in place, it was then time to implement the cars.

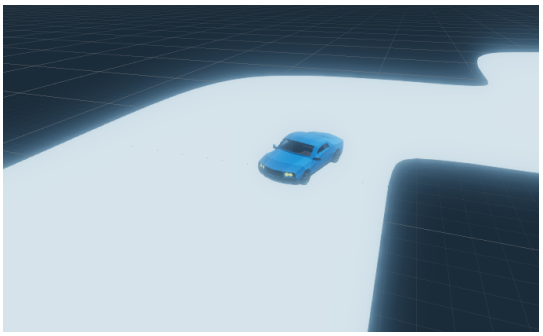


Figure 3.6 The car model on a section of the track

An important factor in making the driving experience feel "right" to the participant was normalizing the controls between the AI and the player. The player's car was created using a realistic physics-based control system, where drag, friction and mass were all taken into account when calculating the forces acting on the car. This allowed the player to drift and sweep through turns to make efficient use of the corner clipping points. Given that the player was able to drift, if the AI did not also have this ability, it may have aroused suspicion if the AI did not also make use of the functionality. For this reason it was decided to omit this functionality all together from the player to both create a fairer playing field as well as lowering the difficulty curve for inexperienced players.

For similar reasons, the 3D car model that was used was the same for both the player and the opponent, so that the player felt they were playing on an even playing field, with neither them or the opponent having a significant unfair advantage.

Given the aesthetic choice of the game was to go with very bright colours, it was also decided that there was a need for an opponent location indicator. This was a small icon that followed the AI car around at all times to give the player an idea of its location if it got too far away.

3.0.3 Testing

2 separate sections were tested during and after the development process, the first was to ensure that the program ran well and maintained a smooth, playable FPS (frames per second) level during game operation as well as ensuring that the different components that make up the game were functioning correctly.

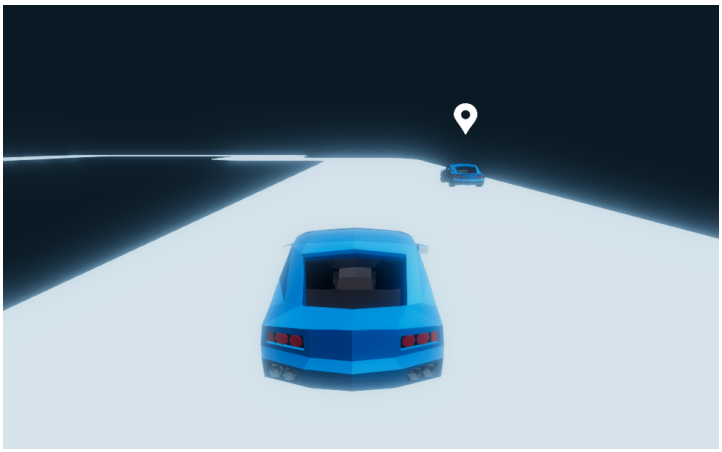


Figure 3.7 The location marker above the AI car

Unity's built-in profiling system allowed for the visualization of CPU, rendering and memory usage as well as the parts that were most resource intensive. The game was tested at the idle position, where the evolutionary training for the opponent had not yet begun and the race had not started. The graph below shows an average frame time of 10ms to render the scene, and an average fps of 100. These values were deemed to be within an acceptable range for optimal playability.

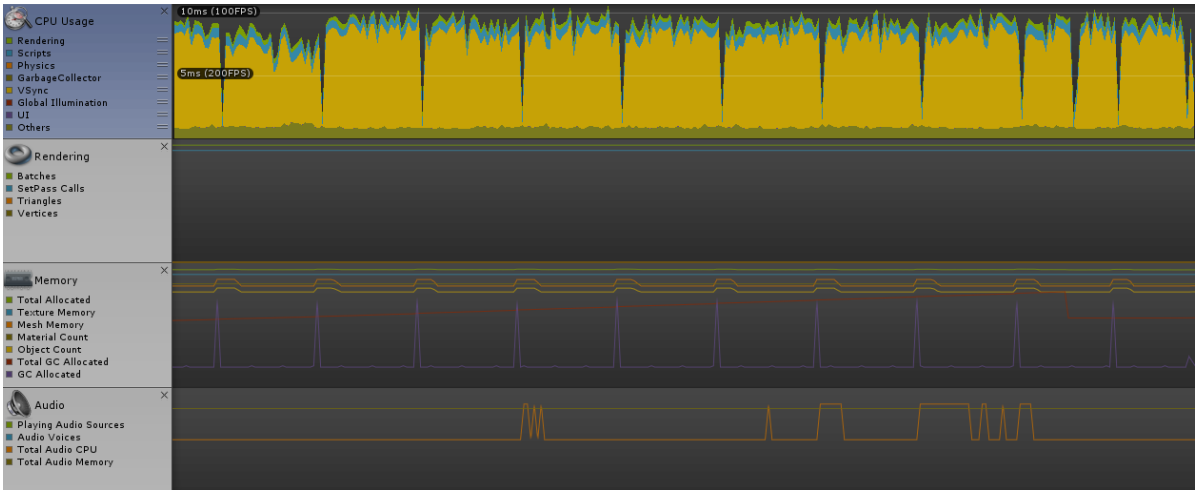


Figure 3.8 Unity profiling graphs before the evolutionary training was initiated

After this, the evolutionary training algorithms were started, and the profiler was run again. A very different graph is shown after this stage, with a large initial drop in FPS while the memory and resources are being allocated for the spawned cars, after which there are intervals of spikes which could be caused by a new generation being created, as each car is despawned and a new set is spawned in their place. While there is a large drop in FPS and average frame

Figure 3.9 Unity profiling graphs after the evolutionary training was initiated

time at this point, it should be noted that this does not affect the performance of the game during the experiment, as the training process is run only once, before the game begins. After the process has complete the performance levels return to normal, even after the trained opponent begins to run.

To test the gameplay aspect of the artifact, each mechanic was tested in turn to see if they were working within the expected operational parameters. The first tests involved tweaking the evolutionary learning algorithm's parameters such as inputs, outputs, speed, sensor range and fitness function output to ensure that the car was behaving as it should.

Several different setups were tested with regards to the inputs and outputs to the neural network. Below is a code snippet from the code that deals with passing the arguments to and from the system.

```
ISignalArray inputsArray = blackBox.InputSignalArray;
inputsArray[0] = frontSensor;
inputsArray[1] = leftFrontSensor;
inputsArray[2] = leftSensor;
inputsArray[3] = rightFrontSensor;
inputsArray[4] = rightSensor;

ISignalArray outputArray = blackBox.OutputSignalArray;

var steering = (float)outputArray[0] * 2 - 1;
```

```
    var acceleration = (float)outputArray[1] * 2 - 1;
    var distance = acceleration * Speed * Time.deltaTime;
    var rotation = steering * TurnSpeed * Time.deltaTime * acceleration
```

In this snippet, inputsArray refers to the array that will store all of the sensory data from the car, and outputArray refers to the array of outputs sent from the final neural network nodes. The 0th element in the outputs array corresponds to the amount that the car should turn in a certain direction, and the 1st element corresponds to the amount of acceleration that should be applied to the car in the forward vector.

The parameter that needed to be adjusted here is the implementation of the normalization function. The output from the network for each node is a value between 0 and 1, and as such, the inputs need to also be given values within this range, and so an activation function is used. There is debate within the field of which activation function gives the best results, for example Specht (2003)[11] wrote a paper on probabilistic neural networks and suggests replacing the commonly used sigmoid activation with an exponential function to see significant performance increases in certain cases.

3.1 Research

Recruiting participants for the experiment proved to be a relatively difficult task. As the subject matter in question was regarding Artificial Intelligence agents within a video game context, ideally participants would have no prior background in these subjects to remove potential bias from the experiment. With this in mind, the participants recruited were from no particular age group with little to no knowledge in AI systems or game development processes.

Once a potential participant was identified, they were given an ethical consent form, which outlines the numerous steps taken to ensure their data is properly handled and kept securely. After they had read the information sheet, they were required to agree and understand the following points:

- I have read and understood the participant information sheet
- I have been given the opportunity to ask questions and have had them answered to my satisfaction
- I agree to take part in this experiment
- I understand that my participation is voluntary and that I am free to withdraw at any time without giving a reason

After having accepted the terms, they could then proceed with the experiment and play the game.

3.1.1 Study Design

When designing the study, the initial hypothesis was that under controlled conditions, participants would not be able to tell if the opponent that they played against was a human or computer controlled. Players would drive around a short racing circuit track against an opponent car controlled by a neural network trained using an evolutionary algorithm. As the experiment was designed around the players individual subjective experience, a mixture of qualitative and quantitative data collection methods were employed. Upon completing the course, players were asked to complete a short questionnaire with several likert scale questions as well as space for them to give self-reported feedback data in the form of a short sentence about what they believed the opponent did or did not perform well at.

Although likert scale data is represented in an ordinal form, results are often treated as interval data to obtain statistical metrics such as standard deviation or normal distribution. In the case of this study however, it would be more beneficial to perform descriptive statistics such as chi-squared tests to get a general feel for the different responses acquired.

The procedure that each study participant experienced was as follows: participants read and agreed to consent to the terms of the experiment. With consent given, they were then given the opportunity to "test run" the game, meaning to drive around the race track for a single lap to get a feel for how the car handles as well as learning the turns of the track. This lap would not be recorded or give any experimental data. After they had completed this lap and felt confident enough to play the game at a competent level, the game was reset and they would play the game once again, this time with the opponent enabled.
An important step in ensuring the validity of the experiment was not telling participants the nature of the opponent they were playing against. If players knew or suspected that the rival car was not a human player, their opinions could become influenced or biased in way that affects the questionnaire outcome at the end of the test.

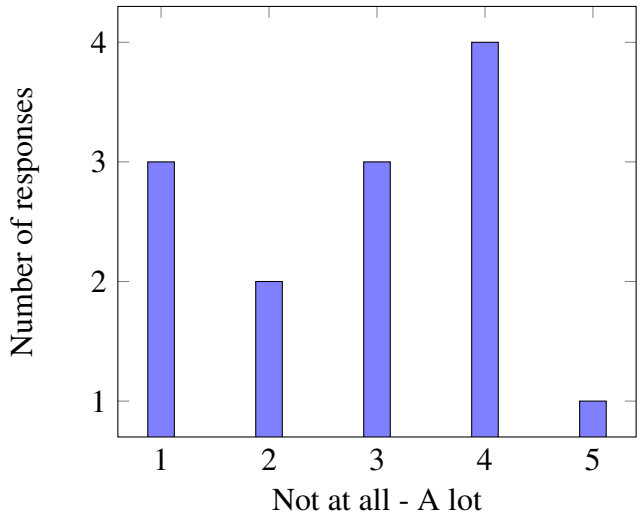
Players were given complete freedom to play the game as they saw fit and experiment with how the opponent handles and reacts to the external stimuli, for example being pushed against the map boundaries, being overtaken or having to make a decision about the direction that it takes around the map.

It should also be noted that the data collected would be valuable regardless of the outcome of the race. The experience gauged would be used in both cases that the players wins or loses. After the lap was completed, participants were asked to complete the questionnaire, with questions based on the work of Laidrd and Duchi (2000) and Wetezl (2004)'s replacement believability criteria for the Turing test, about how the opponent performed in-game such as reaction time, environmental responses, and their perception of how "human-like" they thought the opponent was.

3.1.2 Results

The key answers taken from the participant questionnaire are presented as-is with no additional analysis or interpretation. There were 13 total participants partook in the experiment, who responded to 6 likert-scale questions with scales ranging from 1-5 where 1 is not at all and 5 is a lot. The tables below represent the responses to the individual questions as well as the distribution of values.

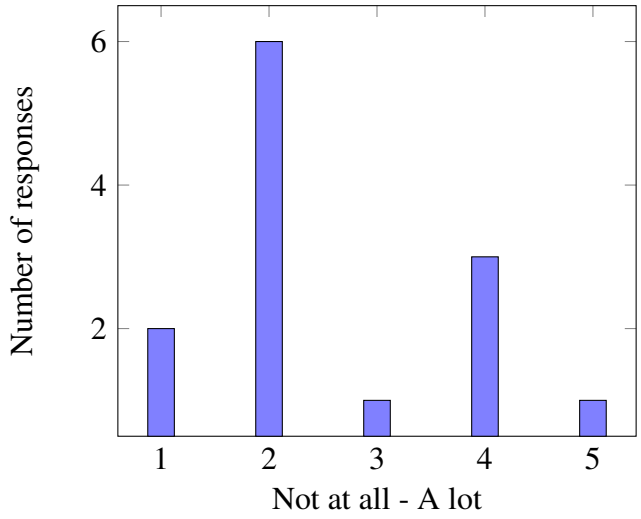
Q1: Did the opponent demonstrate some degree of strategic/tactical planning?



	1 (Not a lot)	2	3	4	5 (A lot)
Number of responses	3	2	3	4	1

Participants were asked to answer if they believed that the opponent they they faced demonstrated a degree of strategy or tactical planning. This could have taken any form from cutting the player off, taking corners at a specific angle or taking a different route from the player.

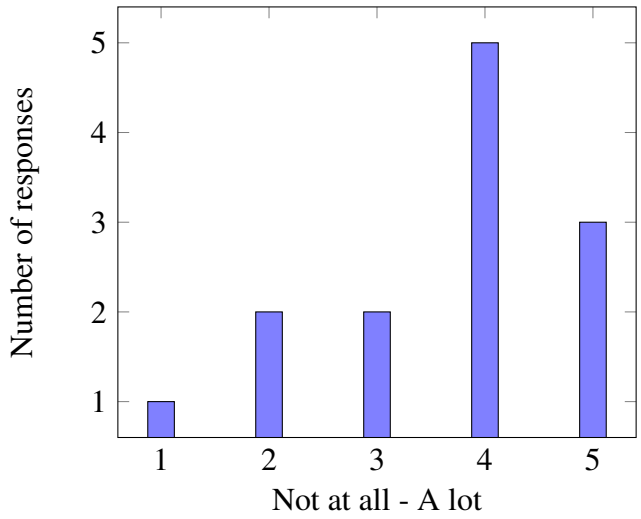
Q2: Did the opponent repeatedly attempt a previous, failed, plan or action?



	1 (Not a lot)	2	3	4	5 (A lot)
Number of responses	2	6	1	3	1

Repeating a previous, failed, plan or action indicates that the AI is not remembering its past encounters and thus not engaging in the learning process. Here, the lower the value the better the player percieved the AI to learn.

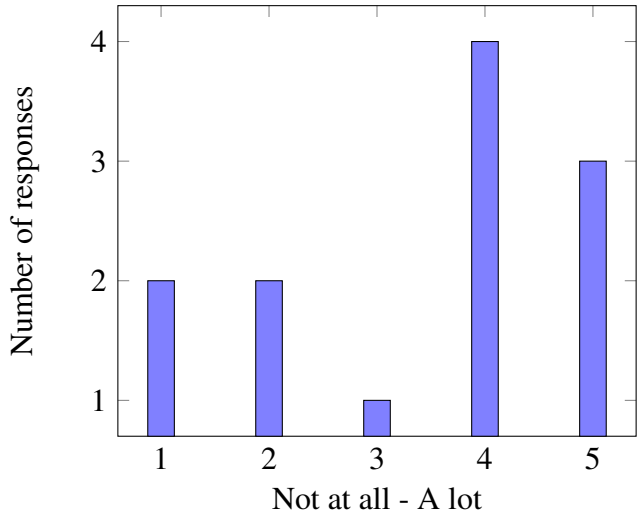
Q3: Did the opponent act with human-like reaction times and abilities?



	1 (Not a lot)	2	3	4	5 (A lot)
Number of responses	1	2	2	5	3

Acting with human-like reaction times is a good measure for how well the AI is "mimicking" human behaviour, for example if the opponent turns too quickly, it will look unnatural to the player and raise suspicion.

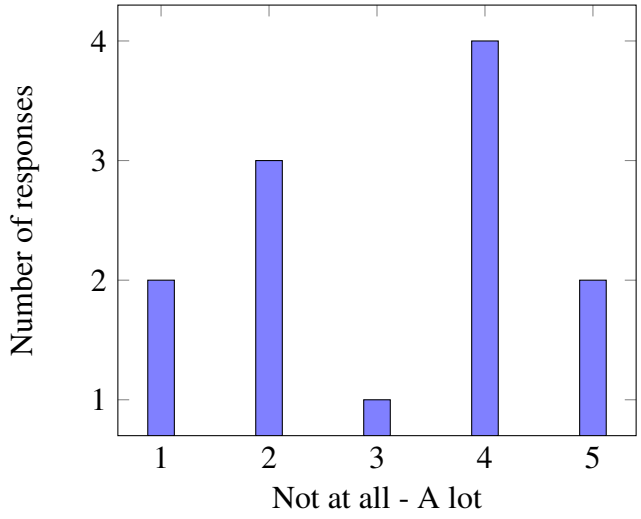
Q4: Did the opponent react to your presence and actions appropriately?



	1 (Not a lot)	2	3	4	5 (A lot)
Number of responses	2	2	1	4	3

Reacting to the player's presence involves the AI making decisions in real time after a player has interracted with it, for example being pushed out of the way or nudged from behind.

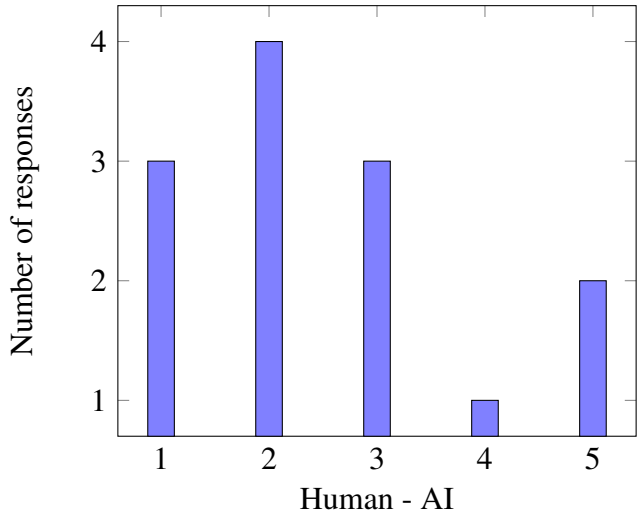
Q5: Did the opponent react to changes in its local environment?



	1 (Not a lot)	2	3	4	5 (A lot)
Number of responses	2	3	1	4	2

Changes in local environment refers to parts of the level that were dynamically adjusted where the AI has to make a decision on the spot about which route to take for example.

Q6: Would you say the opponent was closer to a human or to an artificial intelligence (AI)?



	1 (Human)	2	3	4	5 (AI)
Number of responses	3	4	3	1	2

The final likert-scale question was regarding how, in their opinion, close the opponent was to being either AI or human. The lower this value was the closer they thought it was to being AI.

3.1.3 Results Analysis

In order to interpret and infer meaning from the above results, several statistical calculations can be performed. It should be noted beforehand, however, that likert-scale results are a form of ordinal data, and averaging operations such as mean, to find central tendency are primarily for continuous data types. For this reason, using the mode is the most appropriate method to see frequent responses.

Calculating the sample standard deviation $\sigma = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \overline{x})^2}$ can also be a useful descriptor for the spread of data, where the larger the σ value, the higher the difference in participant opinion.

σ = 1.14

Looking back to the original hypotheses, it was proposed that under controlled conditions, participants would be unable to tell the difference between an AI or a human controlling a race car. Analysis of these results should highlight the different components that make up the AI and their effectiveness in convincing and making participants believe that they were playing against a human.

In the first question, participants were asked to score the opponent's ability demonstrate some degree of strategic/tactical planning. Looking at the distribution chart, there appears to be a slight skewed bias toward the lower end of the scale, indicating that the majority of people felt that the opponent was unable to plan out movements in advance before executing them, and simply played the game base on the data passed to it in real time. This can be considered an expected outcome, as the nature of the opponent's neural network is such that it responds to external input with a weighted output based on the data it was given to train with. No additional memory or planning phases are able to influence the outcome beforehand.

Q2: Did the opponent repeatedly attempt a previous, failed, plan or action?
$\sigma = 2.07$

The second question regarded the opponent repeating an action that a human would deem to be wrong or a mistake. The initial prediction for the outcome of this question was that it would indeed be unable to learn from its past mistakes. Similarly to to the previous question, the opponent is unable to store information about its mistakes given its lack of memory if the learning process before the test did not deem the problem to hinder its ability to increase fitness score and win the game. In this case, the results show a u-quadratic distribution curve, with skews to both sides of the scale, with divided participant opinion about the AI performance.

Q3: Did the opponent act with human-like reaction times and abilities?
$\sigma = 1.51$

The opponent's ability to act with reaction times similar to those of a human is a crucial step in "influencing" the perceptions of participants and maintaining a believable illusion. The results show a clear skew toward the right side of the scale. Participants, for the most part, thought that the opponent responded to challenges such as turning at a rate consistent with a human. As the evolutionary learning process does not require human input, it is surprising that such a convincing illusion was produced, and perhaps demonstrates that little work is needed in this regard as humans will interpolate and make up for its shortcomings.

Q4: Did the opponent react to your presence and actions appropriately?
$\sigma = 1.14$

Reacting to the players presence could be considered the most important aspect of believ-

ability, as with traditional AI agents that are coded to react in a specific way to the player, it can often become too predictable and easy to best, thus the illusion is broken. The graph here shows a large skew to the right of the scale, suggesting the AI was able to react appropriately to invocations from the player such as being pushed or being forced down another path.

Q5: Did the opponent react to changes in its local environment?
$\sigma = 1.14$

Reacting to changes in local environment would indicate to the player a conscious decision making effort from the opponent. The prediction for the outcome of this metric was that it would be able to decide between different choices presented to it, opting for the route which gives the highest fitness outcome. The distribution graph for this question however, shows a split in opinion from participants, with majorities on the left and right of the scale. A possible reason for this outcome is that the opponent may have opted for the same route every time instead of experimenting, or there may have simply not been enough obstacles in the game to get a valid response to this.

Q6: Would you say the opponent was closer to a human or to an artificial intelligence (AI)?
$\sigma = 1.14$

The final 2 questions were open ended feedback questions about the overall performance of the AI. As participants were asked to give positive and negative feedback about their experience, a method is was needed to quantify the data in a meaningful way. A paper regarding contextual polarity in sentiment analysis at a phrase level Wilson (2005) describes a way of determining whether a given expression's polarity is neural or polar as well as providing the amount, if any, of contextual polarity. The combination of all responses were input into the system and are shown below:

- Subjectivity
  Neutral: 0.4
  Polar: 0.6

- Polarity
  Positive: 0.2
  Negative: 0.8

While these results are somewhat rudimentary, it does give an indication of how the participants possibly felt overall. The sentiment analysis reveals that 60 percent of the the feedback gained was polarized toward being positive or negative, and 80 percent of those responded in a negative way toward the opponent's abilities.

This, combined with the final likert-scale question regarding their personal opinion about whether or not they believed the opponent to be a human or AI, provides a conflicting result. The question can be argued to be the most important in gauging how convinced they were of the opponent's nature with the results showing a clear skew toward the left.

Chapter 4

Conclusion

The investigation originally set out to answer the question of whether or not an artificial neural network trained using the NEAT evolutionary learning technique had the potential to be an effective replacement for traditional means of creating AI as a way to create the illusion of intelligence within a video game context. A game which uses the method was created and user participants were invited to play the game, after which they answered questions about the believability of the opponent the faced. The results from this experiment showed that the majority of participants were likely fooled into thinking that they were playing against another human player. Potential reasons for this include the AI reacting with human-like reaction times, reacting to the player's presence and actions appropriately as well as not repeating a previous, failed, plans or actions.

While the experiment showed positive evidence that it's possible to use the NEAT method in such an application, there are several caveats that should be taken into consideration. The first being that the sample size of the participant group was only 13, and a much larger group would be needed to draw a meaningful conclusion. Another factor is the type of game chosen. Racing games can be a good test bed platform to apply the method, however it did prove to lack in ways for the AI to "express itself", and this likely had an effect on the participants perception.

Chapter 5

Reflective Analysis

Looking back on the project as a whole, there were many areas of the investigation which could have seen improvements given additional time. One instance of this is the addition of a human control group during the experiments. While the participants played against AI, having them play against another actual human could have been beneficial in a comparison test to see if the results were merely placebo or if the network provided a real evolution in belivability. Another improvement that I could have made was to create a more complex and engaging environment for both the player and the AI. As I wanted to focus primarily on the AI used than the game itself, little time was spent in the design process which lead to the final game artifact lacking in level diversity as a whole. This addition would have benefited the test results as well in the form of creating a more challenging environment for the AI to show its capabilities.

I would have also liked to create additional games rather than just a racing game, as it doesn't provide a lot of ways for the AI to show personal expression which would have given more depth and thus been even more convincing to the participants.

The theory behind creating a neural network appeared easy at first glance, however problems that weren't anticipated began to arise. An example of this is that the output of the network was queried too fast. This lead to the car having a percievable "jittery" appearance due to the outputs switching back and fourth in an attempt to correct itself.

References

[1] Daniel Livingstone. "Turing's test and believable AI in games". In: *Computers in Entertainment* 4.1 (2006), p. 6. ISSN: 15443574. DOI: 10.1145/1111293.1111303. URL: http://portal.acm.org/citation.cfm?doid=1111293.1111303.

[2] John E Laird, Ann Arbor, and John C Duchi. "Creating Human-like Synthetic Characters with Multiple Skill Levels : A Case Study using the Soar Quakebot". In: (2001), pp. 54–58. URL: http://www.aaai.org/Papers/Symposia/Spring/2001/SS-01-02/SS01-02-012.pdf.

[3] Alabama Ave S Savage. "Step One : Document The Problem". In: (1999).

[4] Fabien Tenc. "THE CHALLENGE OF BELIEVABILITY IN VIDEO GAMES : DEFINITIONS , AGENTS ' MODELS AND IMITATION LEARNING". In: Bates (1992). URL: https://www.enib.fr/%7B~%7Dbuche/article/GAMEON%7B%5C_%7D10.pdf.

[5] Kenneth O. Stanley and Risto Miikkulainen. "Evolving Neural Networks through Augmenting Topologies". In: *Evolutionary Computation* 10.2 (2002), pp. 99–127. ISSN: 1063-6560. DOI: 10.1162/106365602320169811. arXiv: 1407.0576. URL: http://www.mitpressjournals.org/doi/10.1162/106365602320169811.

[6] SethBling. *MarI/O - Machine Learning for Video Games*. 2015.

[7] Matthew Hausknecht et al. "HyperNEAT-GGP : A HyperNEAT-based Atari General Game Player". In: (), pp. 217–224. URL: https://www.lri.fr/%7B~%7Dhansen/proceedings/2012/GECCO/proceedings/p217.pdf.

[8] Marc J V Ponsen et al. "Automatically Acquiring Domain Knowledge For Adaptive Game AI Using Evolutionary Learning". In: (2004), pp. 1535–1540. URL: http://www.aaai.org/Papers/AAAI/2005/IAAI05-012.pdf.

[9] Brandon Koepke et al. "Agile Game Development". In: 1 (2013), pp. 1–8. URL: http://kremer.cpsc.ucalgary.ca/courses/seng403/W2013/papers/05GameDevelopment.pdf.

[10] Colgreen. *SharpNEAT - Evolution of Neural Networks. A C# .NET Framework.* 2018. URL: https://github.com/colgreen/sharpneat.

[11] Specht. "Probabilistic neural networks". In: *Lockheed Missiles Space Company, Inc. USA* (2003).