Mimicking Human Player Strategies in Fighting Games Using Game Artificial Intelligence Techniques

By

Simardeep Singh Saini

A Doctoral Thesis

Submitted in partial fulfilment of the requirements for the award of Doctor of Philosophy of Loughborough University

March 2014

Abstract

Fighting videogames (also known as fighting games) are ever growing in popularity and accessibility. The isolated console experiences of 20th century gaming has been replaced by online gaming services that allow gamers to play from almost anywhere in the world with one another. This gives rise to competitive gaming on a global scale enabling them to experience fresh play styles and challenges by playing someone new.

Fighting games can typically be played either as a single player experience, or against another human player, whether it is via a network or a traditional multiplayer experience. However, there are two issues with these approaches. First, the single player offering in many fighting games is regarded as being simplistic in design, making the moves by the computer predictable. Secondly, while playing against other human players can be more varied and challenging, this may not always be achievable due to the logistics involved in setting up such a bout. Game Artificial Intelligence could provide a solution to both of these issues, allowing a human player's strategy to be learned and then mimicked by the AI fighter.

In this thesis, game AI techniques have been researched to provide a means of mimicking human player strategies in strategic fighting games with multiple parameters. Various techniques and their current usages are surveyed, informing the design of two separate solutions to this problem. The first solution relies solely on leveraging k nearest neighbour classification to identify which move should be executed based on the in-game parameters, resulting in decisions being made at the operational level and being fed from the bottom-up to the strategic level. The second solution utilises a number of existing Artificial Intelligence techniques, including data driven finite state machines, hierarchical clustering and k nearest neighbour classification, in an architecture that makes decisions at the strategic level and feeds them from the top-down to the operational level, resulting in the execution of moves. This design is underpinned by a novel algorithm to aid the mimicking process, which is used to identify patterns and strategies within data collated during bouts between two human players. Both solutions are evaluated quantitatively and qualitatively. A conclusion summarising the findings, as well as future work, is provided. The conclusions highlight the fact that both solutions are proficient in mimicking human strategies, but each has its own strengths depending on the type of strategy played out by the human. More structured, methodical strategies are better mimicked by the data driven finite state machine hybrid architecture, whereas the k nearest neighbour approach is better suited to tactical approaches, or even random 'button bashing' that does not always conform to a pre-defined strategy.

Keywords: Fighting Games, Artificial Intelligence, Finite State Machine, Machine Learning, Game AI, Strategies and Tactics, Mimicking

Acknowledgements

I would like to thank my supervisors Professor Paul Chung and Dr Christian Dawson for their continued support, guidance and mentoring during my research. I would also like to extend my gratitude to Dr Daniel Reidenbach, whose feedback has been of utmost importance in shaping this Thesis.

I am eternally grateful to my mother and father, my brother Amardeep, and my sister-in-law Sarbjit, for believing in me and telling me to never give up. The completion of this work would not have been possible without their undying support, encouragement and patience.

I am indebted to those managers I have worked for over the years who have supported my studies by offering time off work to pursue my research. This has made a big difference and I am grateful for all they have done.

Last, but certainly not least, I would like to thank my friends and family all over the world who have helped me along the way, and whose understanding and patience has been paramount to progressing the doctorate. Many thanks to Jag and Ricky in particular for their constant backing and encouragement over the years.

Above all, I would like to thank God Almighty, without whom this achievement would simply not be possible.

To My Parents

Table of Contents

List of Figures

List of Tables

Part I – Introduction and Background

Chapter 1 – Overview

1.1 Introduction

Many modern videogames are of a competitive nature, either directly with another player in a multiplayer environment, or indirectly via scoreboards in single player games. Traditional multiplayer videogames relied upon players being co-located and playing from the same console or by linking two consoles together. Over recent years, multiplayer videogames are increasingly being supplemented by online play, enabling gamers to either compete or cooperate with one another via gaming services such as PlayStation Network, Xbox Live and Steam. Regardless of how gamers engage one another to play multiplayer videogames, whether it is via an online service, or whether the players are co-located, both players must be available to play the videogame. This restriction presents a challenge and opportunity to assist gamers in their multiplayer experience by enabling them to play against mimicked versions of their human opponents.

The purpose of this Thesis is to establish the current state of research related to mimicking human players in fighting games, and to propose, develop and evaluate an Artificial Intelligence (AI) technique that supports this functionality.

1.2 Motivation

Multiplayer fighting games, where two players fight against one another using martial arts moves in an effort to defeat the opponent, are becoming increasingly popular, especially with the increase of online gaming. The challenge offered to players by playing against other humans is unique in that the opponents use different strategies and tactics, thus offering fresh styles of play. Modern fighting games offer players the freedom to play the videogame using their own strategies, by offering a variety of moves and multiple parameters that are tracked throughout bouts. Since the early 1990s, commercial fighting games have shown far greater diversity by offering fresh game mechanics in an effort to deviate from the standard one-on-one single parameter affair. *Street Fighter Alpha 3* (Capcom, 1998) makes use of three parameters per fighter: health, the block gauge, and the super combo meter. The health of each fighter depletes as they incur damage, however, this can be mitigated by evading the opponent's attacks all together, or blocking them. For every successful attack that is blocked, the block gauge depletes, but refills over the course of time. If several attacks are blocked consecutively, the player is penalised for over blocking and the capacity of the block gauge drops. The super combo meter fills as the player both incurs and deals damage. Once this meter is full, the on-screen fighter is able to execute powerful moves that incur a large amount of damage. These

features (sometimes referred to as game mechanics) allow players to fight strategically, forming their own style and using it against their opponents. This thesis addresses the research question 'How can existing Game Artificial Intelligence techniques be used to successfully mimic human player strategies in strategic fighting games?'

1.3 Research Aims and Objectives

The overall aim of this project is to answer the aforementioned question by researching, proposing, implementing and evaluating an AI system capable of mimicking basic strategies of human players in fighting videogames. The specific objectives are to:

1) Gain an understanding of the current state of research of AI with regards to videogames, as well as its limitations. This is achieved by conducting a literature survey on AI and its usage in modern videogames.

2) Identify and develop an AI solution capable of recognizing long-term player strategies from a human player, and then mimicking these strategies. The solution would be deployed in a multiplayer fighting game, where two humans would play each other, with the AI agent learning the strategies of a given player. Once the AI agent has learned the strategies utilised by a human, it should be capable of using these strategies whilst playing against a human.

3) Evaluate the effectiveness of the AI solution that has been proposed and implemented. Upon designing the solution, it must be implemented as part of a fighting game with a suitable set of rules that lends itself to analysis and evaluation of player and AI fighter strategies. The fighting game must allow players to play in a variety of different ways and incorporate different long-term strategies based on their individual strengths and styles of play.

The method used for evaluation is to compare statistics recorded during a game between two human players with those from games between the same human players and against the AI fighter (mimicking their human opponents from the first set of statistics). The statistics would include the nature and timing of the moves performed by each character, along with the gameplay statistics, such as health etc. These data can be collated and compared to measure the effectiveness of the mimicking technique, and ultimately allow for building a strategic picture (detailed in Chapters 6 and 7).

1.4 Contribution of the Thesis

The problem domain of mimicking humans in fighting games is uncharted territory in terms of academic research. Limited research has been conducted in improving the ability of the AI controlled

fighter, rather than mimicking a human. The major contribution of this thesis are the design, implementation and evaluation of the two AI techniques presented.

The first solution utilises data mining in conjunction with K nearest neighbour classification to solve the problem in a novel way. The solution works by analysing data from a bout between two human players and then using K nearest neighbour classification in real-time during a bout against a human player. K nearest neighbour classification is used to identify which action the AI controlled fighter must take in accordance to the data analysed between the two human players, the current parameters of the game at a given point in time. Details of the design and implementation are provided in this thesis, as well as statistical data from bouts between the AI controlled fighter and the human player, which analysed and evaluated.

The second solution builds on the first, but leverages data driven finite state machines, hierarchical clustering in conjunction with the aforementioned k nearest neighbour classification to generate a novel hybrid AI system architecture. This technique uses a new algorithm that executes a number of steps to identify overarching strategies over numerous bouts, and then uses hierarchical clustering to generate a data driven finite state machine, which is executed by the AI controlled fighter during a bout against a human opponent. Although existing techniques are used, their usage in terms of the problem context is new. The design, implementation and evaluation of both of these techniques form the major contribution of this thesis.

A proof of concept kickboxing fighting game was implemented to evaluate each of the two solutions. The results show that overall, across a sample of ten different strategies, both solutions were equally effective. However, each solution had its own strengths and weaknesses. The design and creation of the proof of concept game as a test bed for Game AI research is a further contribution of this thesis.

The solutions presented in this thesis could be adapted for a variety of game genres, other than fighting games. This would enable players to play against their rivals without directly involving them, as the AI would be mimicking their human opponent, giving players the opportunity to analyse their opponent's strategy and practise against them, much like a boxer would analyse their opponent's past matches, providing them with the advantage to figure out how to counter such strategies. Potential game genres that the two solutions could be applied to include football games, role-playing games and other massively multiplayer online (MMO) games.

1.5 Structure of the Thesis

The remainder of this Thesis is split across three parts, each containing separate chapters. The following breakdown provides an overview of the remaining chapters.

Part I – Introduction and Background

Chapter 2 (Anatomy of a Fighting Game) provides an overview and evaluation of what constitutes a fighting game. Various terms synonymous with fighting games are defined and elements within fighting games are described in detail. Chapter 3 (Strategies and Tactics) provides background on strategy and tactics in the context of videogames and martial arts - two areas that are of particular interest in this Thesis due to their ties with fighting games. Chapter 4 (AI Techniques applied to Fighting Games) introduces and discusses established Game AI techniques. Instances of their application to fighting games are discussed and evaluated. A gap analysis is summarized in terms of the problem domain.

Part II – Analysis, Design and Development

Chapter 5 (Proof of Concept Game) details the proof of concept game that is used as a test bed throughout this research. The details found in this chapter help contextualise the design and implementation of the solutions discussed further in this thesis.

Chapter 6 (K Nearest Neighbour Solution Architecture and Design) presents the K nearest neighbour solution system architecture and detailed design, as well as the rationale for design decisions, and the overall approach that led to the solution. The means by which the K nearest neighbour solution is implemented are also discussed in this chapter. Chapter 7 (Data Driven Finite State Machine Solution Design) presents the solution architecture and detailed design for the data driven finite state machine based solution. Rationale for design decisions and the overall approach that led to the solution are discussed, as well as the means by which the data driven finite state machine solution is implemented.

Part III – Evaluation and Conclusion

Chapter 8 (Evaluation) includes details of the evaluation criteria and method. Experiment design, results and discussion of the results are provided. Chapter 9 (Conclusion and Future Work) concludes the thesis and addresses the original Aim and Objectives put forth in this chapter. An overview of what has been achieved is provided, as well as a view on how the research can be built upon moving forward.

Chapter 2 – Anatomy of a Fighting Game

2.1 Introduction

This chapter provides a detailed account of what qualifies as a fighting game, and as such contextualises the research within this thesis. Key attributes of fighting games are discussed and a brief history of fighting games is provided as a means of surveying the evolution of gameplay mechanics over the years.

2.2 What is a Fighting Game?

With regards to this thesis, 'fighting game' refers to 'fighting videogame'. The research carried out here is concerned with videogames of the fighting genre. Esposito (2005) defines a videogame as: "a game which we play thanks to an audio/visual apparatus and which can be based on a story". While this definition holds true for fighting games, there are specific traits found within fighting games that set them apart from other genres.

This section describes the basic traits found in most fighting games, while refraining from discussing mechanics that do not conform to the norms found within traditional fighting games.

2.2.1 The Rules

Typically, a fighting game will always display more than one character on screen and each character has a health meter. Traditional fighting games are 1-on-1, allowing for precisely two characters on-screen. At least one on-screen character is controlled by a human player, with the remaining character(s) either controlled by the computer or other human players (see Figure 2.1 for a screenshot of a typical fighting game). The two on-screen characters must fight each other in hand-to-hand combat (although weapons are factored into some fighting games), each using martial arts moves to defeat the opponent. As fighters successfully connect an attack with their opponent, the opponent's health meter is depleted. The objective is to completely deplete the opponent's health meter, at which point the round is complete and the victor is determined. A fight may last several rounds, with the typical default setting being the best of three rounds. The rounds are time bound, with a clock counting down to zero marking the end of the round. If neither fighter has been defeated when the clock reaches zero, the fighter with the most health is hailed the victor. The timer can usually be altered or switched off within the game options.

2.2.2 The Fighters

Each on-screen fighter has a variety of actions they can perform at the players' behest. The player will be presented with a variety of characters from which they choose one. Each character has their own

unique set of actions, as well as inheriting certain actions that can be carried out by all characters. This common set of actions typically includes the following:

- Traverse the screen towards or away from the opponent.
- Jump directly above, towards or away from the opponent.
- Crouch to evade or perform low targeting attacks against the opponent.
- Perform basic kicks and punches whilst standing, jumping or crouching.
- Perform a grapple or throw action while at extremely close proximity to the opponent.
- Block while standing or crouching (and in some cases while jumping).

In terms of animation, the basic punches, kicks and throws usually vary from character to character, however the net result is always the same in terms of damage dealt and proximity at which the action is effective. The unique actions retained by each character are often referred to as 'Special Moves'. These moves inflict greater damage than standard actions and often display flashy animations, however while standard moves are usually performed by a single button press, a combination of directional and action buttons must be pressed to execute a special move.

More recently, the concept of 'Super Moves' has become common in fighting games. This concept introduces a gameplay mechanic where each fighter has a further meter that is filled or emptied as they inflict and receive damage. Once the meter is full, a complex sequence of action and directional buttons can be pressed by the player to execute a move that deals a substantial amount of damage, beyond that of a special move. Examples of games featuring Super Moves are provided in Section 2.3.3.

2.2.3 The Environment

The environments within which the bouts take place are bounded as the players move to the extreme left or right of the stage. Most fighting games take place on a two-dimensional plane, however, some three-dimensional fighting games allow for sidestepping, adding a third dimension to the movement of characters. Both fighters are displayed from the third-person viewpoint, with their profile visible to the player. Depending on the specific videogame, the fighters are usually facing each other. When one fighter jumps over their opponent, the opponent shall automatically turn around, resulting in the switching of sides. In fighting games such as Bandai Namco Games' *Tekken Tag Tournament 2* (2012), the opponent remains with their back facing the fighter that jumped. In this particular videogame, this enables the player facing the opponent's back to carry out a different set of actions.

2.3 A Brief History of Fighting Games

Having established the typical traits of a traditional fighting game, this section explores the history of fighting games and the evolution of their mechanics. The section is divided into major milestones in fighting game history, with innovations and evolutions in design described along the way.

2.3.1 Early Fighting Games

The earliest example of a videogame resembling something close to a traditional fighting game is *Heavyweight Champ* by Sega (1976). This videogame presented the player with a side view of two on-screen fighters, each with their own health meter. However, at the time, this fighting game did not rely on traditional button based control inputs. Instead, it utilised a glove peripheral built into the arcade cabinet, which could be moved up and down to vary where attacks were targeting. It was the likes of *Karate Champ* (Technos Japan, 1984) and *Yie Ar Kung-Fu* (Konami, 1985) that popularised the genre and offered a greater variety of moves. However, these early fighting games only featured one playable character for single player bouts against the computer fighter, and there was no notion of special moves. *Street Fighter* (Capcom, 1987) changed this by embedding secret special moves, the button combination for which would need to be deciphered by the player. Street Fighter included many of the basic traits synonymous with fighting games such as multi-player functionality to allow two players to fight each other, definitive round timers, and a multitude of standard and special moves.

2.3.2 The 16 Bit Era

In 1991, Street Fighter spawned a sequel, *Street Fighter II* (Capcom, 1991) which revolutionised fighting game genre. Street Fighter II presented players with an option of choosing from eight playable characters, each with their own unique set of special moves. The standard move set was far greater than any fighting game that preceded it with a total of six attack buttons, each giving rise to a unique animation. In some instances, these standard moves gave rise to multiple variations of the same move when combined with a directional button press. For example, pressing forwards and the attack button would have a different animation to pressing the attack button in isolation. The level of variety offered by having multiple characters to choose from, each with unique special moves and various flavours of standard moves, coupled with the cutting edge graphics, fluid gameplay and competitive appeal made Street Fighter II the benchmark for fighting games in the early 1990s.

Shortly after the release of Street Fighter II, another noteworthy fighting game was released. *Mortal Kombat* (Midway, 1992), built upon the groundwork put forward by Street Fighter II, sparked great controversy due to its use of animated blood which was perceived as being excessively violent at the

time, but has since become a staple feature of many fighting games. Mortal Kombat also introduced the notion of a 'finishing move'. Once the bout is over, the victor is given the opportunity to execute their opponent by inputting a specific button command and trigger a unique finishing move, referred to as a fatality within the videogame. While the use of finishing moves has been largely localised to the Mortal Kombat videogame series, this idea may have given rise to the notion of further unique powerful moves. These moves would eventually be called Super Moves and if executed would deal the finishing blow.

2.3.3 The Introduction of Multiple Parameters

Super Moves first appeared in *Super Street Fighter II Turbo* (Capcom, 1994). This fighting game built on previous versions of Street Fighter II, featured a Super Combo Meter that filled up as the player executed special moves and incurred damage upon their opponent. Once the meter is full, a specific button command can be pressed to unleash an exceptionally powerful move, known as a Super Move or Super Combo. If the Super Combo deals sufficient damage to defeat the opponent, the screen fills with flashes of light making for a rewarding means of defeating the opponent.

Super Street Fighter 2 was the first time a fighting game used a meter beyond the standard health meters. Figure 2.1 shows a screenshot of Super Street Fighter 2, with the health bars for each character situated at the top of the screen, and super combo meters situated at the bottom of the screen.

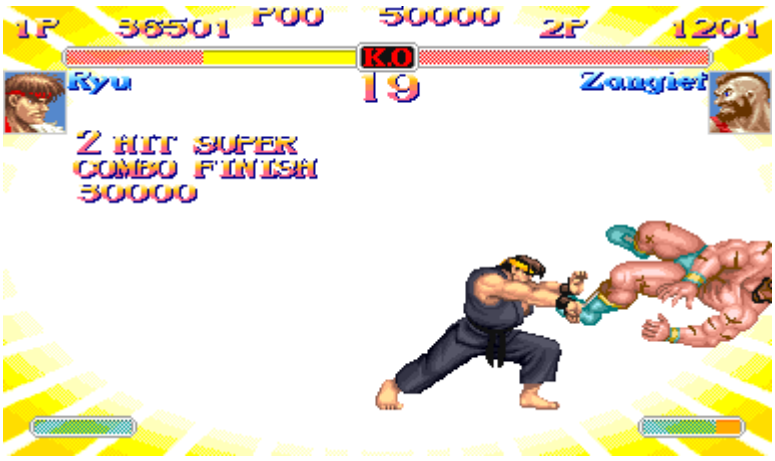

Figure 2.1 – Super Street Fighter 2 Turbo Screenshot

This notion of using multiple parameters was expanded upon over the years with *Mortal Kombat 3* (Midway, 1995) using a 'momentum meter' that filled up over time and allowed players to run and execute combinations. *Mortal Kombat Trilogy* (Midway, 1996) used a further meter called the

Aggressor bar, which filled up as the player executed attacks. Once the bar was full, the player could activate it, enabling them to move faster and inflict greater damage.

Street Fighter Alpha 3 (Capcom, 1998) went even further in terms of integrating multiple parameters. Beyond the standard health meter, there were three separate variations of the Super Combo Meter, one of which allowed the player to execute custom combos similar to Mortal Kombat Trilogy's Aggressor Bar. Further to the Super Combo Meter, each player has a block gauge, which fills up rapidly over time. As the player blocks successive attacks from their opponent, the gauge depletes. When the gauge is empty, the player's defence is shattered and they are left temporarily vulnerable. The gauge then fills up again, but the capacity is reduced each time it is shattered. The intention of this mechanic was to penalise players who are too dependent on blocking, rather than stopping the opponent's attack mid-flow, or evading the attack altogether, both of which take more skill than blocking. This led to a new element of strategic play within the fighting game genre.

2.3.4 Three Dimensional Fighting Games

Early and subsequent fighting games utilised two-dimensional graphics. This was largely due to the limitations in hardware. Even in the 32-bit era (post 1994), many fighting games, such as *Street Fighter Alpha* (Capcom, 1995), still focused on delivering a two-dimensional experience as this was an integral part of their identity. The introduction of 32 bit consoles and superior arcade hardware led to three-dimensional fighting games where fighters were portrayed by 3D character models, rather than hand drawn sprites, traversed an environment rendered in 3D. However, the majority of early 3D fighting games restricted movement of the 3D environment to a 2D plane. *Battle Arena Toshinden* (Tamsoft, 1994) introduced the ability for fighters to move along X, Y and Z-axes, allowing players to dodge projectile attacks. Figure 2.2 shows characters moving across the 3D plane in Battle Arena Toshinden.

Figure 2.2 – Battle Arena Toshinden Screenshot

Three-dimensional movement within fighting games took off with *Tekken 3* (Namco, 1997) and *Dead or Alive* (Team Ninja, 1997) following suit. However, while certain fighting franchises have made the move to 3D graphics, the characters are still rooted in two-dimensional planes as a conscious design decision rather than a limitation of the hardware. Such videogames include *Ultra Street Fighter IV* (Capcom, 2014), which is shown is Figure 2.3 below, *Mortal Kombat* (NetherRealm Studios, 2011), *Street Fighter X Tekken* (Capcom, 2012) and *Marvel vs. Capcom 3* (Capcom, 2011).

Figure 2.3 – Ultra Street Fighter IV Screenshot

2.3.5 Further Enhancements

While traditional fighting games are typically one-on-one, *X-Men vs. Street Fighter* (Capcom, 1996) introduced the notion of tag battles in an otherwise conventional fighting game. Each player would choose two characters to make for a two-on-two battle, where one member of each team would be on-screen doing battle, with their partner waiting to be tagged-in off screen. Each member of the team has his or her own health meter. Once one member of the team is defeated, the other is automatically tagged-in to continue the bout. The victor is declared once both fighters belonging to a team are defeated. This concept was taken further with *Marvel vs. Capcom 2* (Capcom, 2000), where each player could choose three fighters per team. Both videogames also featured the ability to perform tag Super Moves, where each member of the team would perform their Super Move simultaneously, incurring more damage. Other fighting games followed the trend set by Capcom, including *Tekken Tag Tournament* (Namco, 2000) and *Dead or Alive 2* (Team Ninja, 1999).

Additional enhancements in fighting games relate to the environment in which the bout takes place. *Dead or Alive 2* (Team Ninja, 1999) featured hazards in the arena, such as explosions that would incur damage if a character was within the blast radius. This gameplay mechanic added a level of strategy in that it opened up players to use the environment to their advantage. *Tekken 4* (Namco, 2002) followed up on this by providing destructible environments, allowing fighters to corner one another and use the destructible environment to incur extra damage. This is shown in Figure 2.4 below.

Figure 2.4 – Tekken 4 Screenshot

While fighting games should not be confused with a traditional Role Playing Game (RPG), certain fighting games have begun to incorporate elements of RPGs within their own game rules. RPGs are traditionally single player affairs, although they can also be played online with other human players in the form of a Massively Multiplayer Online Role Playing Game (MMORPG) such as Elder Scrolls Online (ZeniMax Online Studios, 2014). RPGs typically follow a protagonist who engages with and speaks to Non-player Characters (NPCs) while a story unfolds. Combat can be either turn-based or real-time and often leverages physical attacks as well as projectile magic attacks. The character is rewarded points for battles won, which in turn improve his or her skills and abilities, enabling them to face adversaries that are more challenging. Examples of turn-based RPGs include Final Fantasy VII (Squaresoft, 1997) and Bravely Default (Square Enix, 2013), whereas real-time, or action RPGs include Diablo 3: Reaper of Souls (Blizzard Entertainment, 2014) and Tales of Xillia 2 (Bandai Namco Games, 2014).

Despite being one of the earliest game genres established, fighting games are still prominent in today's gaming market. Games such as Street Fighter 4 (Capcom, 2008), Tekken Tag Tournament 2 (Bandai Namco Games, 2012) and Mortal Kombat (NetherRealm Studios, 2011) are developing the genre further and paving the way for future entries for each of the respective franchises. Even classic two-dimensional fighting games are making a resurgence in the forms of BlazBlue: Chronophantasma (Arc System Works, 2011) and *Person 4 Arena Ultimax* (Arc System Works, 2014).

2.4 Chapter Summary

In this chapter, a detailed account of what constitutes a fighting game has been provided. A fighting game has been defined as a videogame where two on-screen characters, each with a health meter, must engage in combat using standard and special moves to incur damage, with the objective of depleting the opponent's health meter. The genre of fighting games can encompass features beyond this definition. This chapter has provided an overview of how different features have evolved and become embedded within many commercial fighting games. These features include three-dimensional movement, multi-parameter combat, tag team battles and hazardous environments.

This chapter has given examples of many commercial fighting games, many of which have added their own features to the genre, while maintaining the core principles that define fighting games. Moving forward, these principals shall be used in developing a proof of concept game that shall be representative of the commercial fighting games discussed here.

Chapter 3 - Strategies and Tactics

3.1 Introduction

Strategy can be defined as a preliminary decision making activity, whereas tactics can be defined as an action based decision-making activity (Mouchet, 2005). This definition contextualises strategy and tactics, which is of utmost importance in this thesis. Strategy are long-term approaches to achieve an ultimate goal, whereas tactics are short-term gains that facilitate the strategy. In this chapter, strategy and tactics as they pertain to martial arts in popular media shall be explored. Observing martial arts movies helps contextualise the use of strategy and tactics as they are used in fighting games. This chapter also reviews literature on strategies and tactics, as it relates to videogames, both commercial and academic, as well as its place in the world of martial arts. Views on the differences between strategies and tactics are put forward. This is undertaken in an effort to better understand, and place context around the aims and objectives of the Thesis.

3.2 Strategies and Tactics in Martial Arts

To understand how strategies and tactics could be used in fighting videogames, it is important to consider their use in martial arts, as this is the real life domain that is being replicated in fighting videogames. Two ancient texts that still bare much relevance today with regards to martial arts strategies are The Art of War (Sun Tzu, 1910) and The Book of Five Rings (Musashi, 1643).

The Book of Five Rings (Musashi, 1643) is a tome covering martial arts strategy and is split into five sections, or 'books'. The author, Miyamoto Musashi was a Japanese swordsmen and ronin (master-less samurai) who came to prominence on account of his numerous sword fights during the Edo Period, which succeeded medieval/feudal Japan.

The first book, The Book of Earth, focuses on Musashi's own strategy, which he taught to his students at the Ichi School. In The Book of Earth, Musashi regards strategy as knowing when to use an appropriate weapon, and placing great importance on timing. Musashi discusses the usage of spears, longs swords, short swords and even guns. Musashi stresses that a single weapon should not be overused, and that variety in using weapons is a critical success factor when engaging an opponent. This principal can be extended to hand-to-hand combat in general where one could argue that relying on the same set of moves in continuous rhythm makes for a predictable outcome, largely because the opponent will grow accustomed to the move patterns, making it easier for them to spot and exploit a weakness. Another core principle in The Book of Earth is the importance of timing in combat situations. Musashi contemplates timing by stating it is paramount to all walks of life, not just combat.

He then goes on to state that a warrior must know when to attack and when not to attack based on what is happening around them in the background, as well as what is happening immediately in-front of them.

The second book, The Book of Water, focuses further on strategy. The name was quite possibly chosen based on the view that a warrior must be able to shift from using one set of tactics to another with fluidity, balance and flexibility. This book also focuses on the spiritual side of martial arts, emphasising the importance of staying calm while executing a strategy, as well as factoring into account your opponent's frame of mind. This is fundamental to fighting as an opponent's rage or fear could be used to a skilled fighter's advantage. Perception and peripheral vision are also regarding as tools to be leveraged when defining and executing a strategy. Musashi cites one's ability to fight an enemy directly in front of him/her, while scouting for reinforcements in the background as being one such use of leveraging peripheral vision to aid combat. The Book of Water goes on to describe various other techniques and tactics that would be used within a strategy. Namely, Musashi goes over the five 'attitudes' to swordsmanship, which are based on anatomical regions that are used as targets, when one should attack in a certain region. Other tactics Musashi refers to including feinting and biding ones time to find an opening.

The third book, The Book of Fire, covers strategy further, but rather than focusing on the techniques that are featured in The Book of Water, which are arguably tactics, this book focuses more on the warrior leveraging the environment and opponent's weakness to their advantage. The chapter places importance on preparation ahead of the battle in terms of having the correct armour, knowing the terrain and surrounding area and knowing the opponent's disadvantages. This book is particularly interesting as it further enforces the notion that strategy must be pre-defined, with clear objectives and contingency plans in place.

Of the five books, the first three focus on the broader context of strategy. The fourth book, The Book of Wind, places further importance on knowing the opponent, but serves primarily to explore techniques that were being used by other schools at the time, making it of little relevance to today's martial artists. The final book, The Book of Void serves as an epilogue and discusses spirituality at greater lengths.

While historians debate the exact year it was first published, Sun Tzu's The Art of War was first translated into English by Lionel Giles in 1910. The Art of War (Sun Tzu, 1910) details the methodology to formulating a strategy before engaging an enemy in the battlefield, and like The Book of Five Rings (Musashi, 1643), discusses the numerous considerations to be made in executing the strategy. The

book is split into thirteen chapters, each covering a major theme of warfare. While the text focuses on military tactics and strategy, its teachings have been applied by many in the world of corporate business.

Many of the themes across the chapters of The Art of War (Sun Tzu, 1910) fall in line with Musashi's own considerations as documented in The Book of Five Rings (Musashi, 1643). The book advocates making strategic and tactical decisions by placing great importance on terrain, surroundings, knowing the enemy and knowing a unit's own military strength. Emphasis is placed on attacking quickly and ending the battle as soon as possible where applicable. The book advocates forward planning and avoiding confrontation where possible, offering tactics on how this can be achieved. One of the chapters titled 'The Nine Battlegrounds' provides nine conflict scenarios and provides details on what the commander would need to consider, as well as the ideal strategy for being victorious in each of the scenarios. While The Book of Five Rings (Musashi, 1643) focuses more in individual conflicts such as a swordsman facing one or more assailants, The Art of War is primarily focused on larger scale engagement such as armies going to war, and is primarily authored as a guide to the commander. However, its teachings can certainly be extended to one-on-one combat or even conflict resolution.

While the ancient text of The Art of War (Sun Tzu, 1910) and The Book of Five Rings (Musashi, 1643) provide a primer for strategic and tactical considerations, they can be difficult to put into context of the modern one-on-one competitive martial arts that relate to fighting games. One such martial art that encourages strategic engagement is Muay Thai. Van Schuyver and Villalobos (2002) discuss various Muay Thai fighting strategies and ultimately suggest that nearly all fighters can be categorised into one of four fighting styles; aggressive, counter, elusive and tricky. Each of these types of fighters has an underlying strategy. The aggressive fighter favours a brute force approach, relying on strength and a good offense. The counter fighter adopts a strategy of waiting for their opponent to make the first move then capitalising on any openings or other opportunities. The strategy adopted by the elusive fighter is to move around such that the opponent is chasing and retreating from them as they dictate their opponent's actions through swift movement. Finally, the tricky fighter is one that relies on faints and deception, as well as intimidation. In addition to these fighting classifications, range strategy is used to determine when particular types of moves should be executed (Van Schuyver and Villalobos, 2002). Turner and Van Schuyver (1991) argued that 97% of all of the world's martial artists could be classified into one of these groups. By identifying which class an opponent belongs to, it is argued that strategies can be developed to defeat them by exploiting this knowledge. This falls in line with Sun Tzu's Art of War, where emphasis is placed on the notion of observing and knowing the opponent's own strategy and tactics, and launching an attack against it. While Sun Tzu's approach

appears evident in the work put forward by Van Schuyver and Villalobos (2002), it should be noted that the original work of Sun Tzu is applied in the context of military tactics. Only after strategies have been defined can tactics be derived. As Van Schuyver and Villalobos (2002) suggested, strategies must be derived before entering the ring (prior to the fight) and tactics should be executed and adjusted during the fight. In some instances, it may be necessary to adjust the strategy as more is learned about the opponent. Musashi (1643) highlighted timing as being of the utmost importance for strategy, and while this is undoubtedly true, the same, if not more importance could be placed on the timing of tactics as these are shorter term.

Lee (1975) referred to tactics as the brainwork of fighting and stated that they are based on observing and analysing one's opponent before making intelligent choices on how to engage them. As Lee (1975) put it, the use of tactics (also referred to as tactical fighting) consists of a three-part process; preliminary analysis, preparation and execution. Preliminary analysis is the observation phase where a fighter scrutinises their opponent to notice any bad habits or traits that can be exploited. The preparation phase may consist of lining the opponent up for an attack by controlling movement into a certain position, or by carrying out feints. The execution phase is where the attack is carried out. Lee (1975) emphasises that this must be done in continuous motion without hesitation and that timing is of great importance. Lee's (1975) three-phase process is particularly interesting as it can be easily modelled by an AI for use in a fighting videogame. For example, prior to a bout between an AI and human player, data analysis could be performed to understand the human player's weakness. The AI may then attempt to re-create the moves that precede the weakness in an effort to exposing it during the bout itself. Should the weakness expose itself, machine learning or even an FSM could be used to instruct the AI player to execute a move to exploit the weakness. Both of these techniques are discussed in Chapter 4 in greater detail.

3.3 Martial Arts Movies

Martials Arts has garnered much attention due its exposure in the movie industry. As a result, many fighting games pay homage to martial arts movies by including parodies of iconic characters or paraphernalia in the game, from Bruce Lee's yellow jumpsuit (Game of Death, 1978) which was included in Tekken 3 (Namco, 1997), to Han's claw from Enter The Dragon (1973) which inspired Vega's claw in Street Fighter 2 (Capcom, 1991). While not always realistic, martial arts movies can provide a view as to how strategies and tactics can be used to overcome imbalances when facing different opponents. This is a theme that has been projected in countless martial arts movies since

their rise to prominence, often showing that a smarter fighter always succeeds, even if his/her opponent is stronger and faster.

Martial arts movies were popularised in the West following the release of Enter the Dragon (1973), starring Bruce Lee as the movie's protagonist. The films plot followed Lee's character, also named Lee, infiltrate a secret martial arts tournament held on an island by a disgraced former shaolin monk, Han, who was the head of a global criminal empire specialising in drugs and arms trafficking. Lee was charged with defeating Han and bringing an end to his empire, avenging the death of his sister, who was apparently murdered by Han's henchmen. While the plot in itself seems cliché by today's standards, it was the action and fight scenes that brought the movie to the attention of critics and cinemagoers alike. The climatic final battle between Lee and Han saw Han leveraging a hall of mirrors to his advantage, leaping out and striking a disoriented Lee, before going back into hiding to plan his next attack. After a few hits, Lee becomes wise to Han's strategy and counters by leveraging his own strategy; smashing the mirrors in the hall and then attacking a vulnerable Han.

The concept of having the protagonist outsmart their opponent by learning and countering their strategy has been included in many martial arts movies since Enter the Dragon (1973). In the original cut of Bruce Lee's Game of Death (1978), Lee faced a tall and strong opponent played by then NBA basketball star, Kareem Abdul-Jabbar. Abdul-Jabbar's character is sensitive to sunlight, forcing him to fight his opponents in a darkened room. Lee is knocked down several times by Abdul-Jabbar's long reach and powerful strikes, but then parries the moves to get close to his opponent, where he incurs damage to Abdul-Jabbar's rib cage and mid-section, which are roughly in line with Lee's line of sight. Using his opponents towering height and predictable strategy to his advantage, Lee is able to defeat Abdul-Jabbar.

With the passage of time, and the rising popularity of martial arts, movies are incorporating new fighting styles and scenarios to emphasise the importance of strategy. Modern martial arts movies such as The Raid (2011), The Raid 2 (2014), Man of Tai Chi (2013), The Protector 2 (2014) and Badges of Fury (2013) all feature climatic battle sequences at the end where the protagonist leverage their cunning to counter the opponents seemingly unbeatable strategy, leading them to victory. In The Expendables (2010), Dolph Lundgren fights Jet Li in hand-to-hand combat. Lundgren utilised his size and strength in conjunction with Western fighting techniques based on kickboxing, whereas Li, who is significantly shorter than Lundgren uses Chinese Wushu and his environment to his advantage. When the two fight, Lundgren uses the reach of his punches to keep Li at bay, while Lee is struggling to strike

Lundgren's head on account of his height. Li deliberately leads Lundgren to a scaffolding area where Lundgren must duck to resume the fight, bringing him to Li's level, evening the odds.

Donnie Yen's portrayal of the legendary Wing Chun practitioner, Ip Man in the movies Ip Man (2008) and Ip Man 2 (2010) have made him renowned as being one of the most prolific Asian film stars in the world. In the movie Ip Man 2 (2010), Ip Man must do battle with a western heavyweight boxer, played by real life English martial artist Darren Shahlavi, in a bout where Ip Man cannot kick. During the first few rounds of the boxing match, Ip Man falls prey to the bigger, better-conditioned boxer's strikes, and struggles to inflict any real damage. Due to the high level of conditioning boxers undergo, Ip Man's punches inflict little damage, and the boxer maintains his level of stamina. Following the first few rounds, Ip Man changes his strategy and precision targets the boxer's biceps, putting his arms out of commission, preventing him from dealing damage, but also blocking Ip Man's rapid succession of punches. Ip Man then targets the boxer's rib cage, knowing that this area is vulnerable to even the most conditioned fighter. Ip Man then launches a full assault of rapid punches, ultimately defeating the boxer who can no longer fight back or defend himself. This type of strategy adjustment is also seen in Jet Li's critically acclaimed Fist of Legend (1994), which is in itself a remake of the Bruce Lee classic, Fist of Fury (1972). During the final fight of the movie, Li's character Chen is fighting a Japanese General, whose usage of techniques is far more proficient than Chen's. Leveraging knowledge imparted to him earlier in the movie, Chen changes his style mid-fight to use more westernised kickboxing techniques that the general is not accustomed to. This ultimately leads to Chen's victory.

3.4 Strategies and Tactics in Videogames

An important element of the research conducted in this Thesis is differentiating tactics from strategies, and how they are applied in to videogames. In the context of videogames, strategies are defined as higher-level goals of some effort that are fulfilled by meeting tactical objectives (Hildmann and Crowe, 2011). This is to say that while strategies are long-term approaches to achieve an ultimate goal, tactics are short-term gains that facilitate the strategy.

One-on-one fighting games, such as *Street Fighter 2* (Capcom, 1991), do not lend themselves to in-depth strategic elements of gameplay (Gingold, 2006). Gingold argued that fighting games like Street Fighter 2 are little more than variants of Rock, Paper, Scissors (RSP). Gingold presented a number of variants of RSP, with each variant moving closer to the basic mechanics found in a fighting game. While this argument may hold true for 'button bashing' play tactics (where two humans play each other and randomly press buttons on a control pad without knowing the consequences of the button press), one may argue that a skilled player can anticipate their opponents next move based on the previous

moves. Further to this, a skilled player may stand idle and react to their opponents move once it has been made, rather than performing a random move simultaneously. While not of the same genre as fighting games, Real Time Strategy games feature tactical and strategic decision making to a high degree. Real Time Strategy games are videogames where players command a group of resources, human and machine, to fight against other groups of enemies. Players need to rather sustenance, build structures, train their fighters and conduct warfare (Liu et al, 2014). Robertson and Watson (2014) focus their research on Real Time Strategy games and regard tactical decisions as being short term objectives such as eliminating enemy firepower in a the shortest possible time. The action to perform this would be considered a tactic, whereas strategy is regarded as being longer term. Robertson and Watson (2014) state that strategic decision making in real time strategy games is based on knowing the sequence of actions to take to achieve a long term goal.

Robocode has been at the heart of much videogame related AI research. Robocode is a videogame where a tank coded by the user has to make moves in a battlefield where it evades the opponent's attacks whilst attacking the opponent itself. There are many factors to consider when coding the tank such as the nature of its opponents, use of energy and movements against opponents. The tank loses energy as it fires bullets, but gains energy upon successfully attacking an opponent (Hong et al, 2004).

Hong et al (2004) discussed the limitations of existing techniques in developing 'extra' behaviours; behaviours that have not been manually designed by a human, but rather are 'emergent' from a set of low-level actions. This notion is not dissimilar to the contrast between strategies and tactics. Hong references Artificial Life, the study of synthetic systems to show the behaviour characteristics of natural living systems (Johnson and Wiles, 2001) as an example of an emergent behavioural system. Artificial Life uses a set of low-level behavioural actions to create a high-level complex and whole behaviour (Hong et al, 2004).

Hong, et al (2004) uses Robocode to create primitive behaviours based on actions that are to be carried out within that videogame, which are then classified into a variety of strategies. An optimal composition of primitive behaviours is selected based on the current state of the game. The encoding of the overall behaviour is split into six strategies; move, avoid, shoot, bullet power, radar search and target select. Each of these strategies encompasses a set of low-level behaviours. The behaviour of a tank is encoded as a composition of behaviours from various strategies and is selected.

3.5 Chapter Summary

This chapter provides an account of current literature related to strategic and tactical execution within martial arts and across a variety of academic videogames. Martial Arts movies are also consulted to help put the notion of strategies into context of one-on-one unarmed combat. The distinction between strategies and tactics is clarified in the context of fighting as well as gaming. This chapter has contextualised the usage of strategies and tactics, while identifying further gaps in current research. Generally, strategies can be considered long term plans to achieve a goal, and are established at a high level. Tactics fall within strategies, and provide the detail on how a given strategy shall achieve its goal. In martial arts, a finite number of strategies exist and different tactics are implemented to facilitate the strategy in achieving victory. Strategies and tactics are applicable to many videogames including Street Fighter 2 and the academic game, Robocode. Again, the strategies are identified as being longer-term approaches to achieve a goal, such as defeating an opponent in Street Fighter 2, whereas the tactics are the means by which this is accomplished. In the context of Street Fighter 2, this could be a case of standing idle and countering the opponents offensive moves.

Having established the key difference between strategies and tactics, as well as their usage across videogames and martial arts, further clarity has been established pertaining to the problem domain. The focus of this thesis is to mimic human strategies in fighting games, which encompasses all tactics and operations. Now that a firm understanding regarding what constitutes a strategy, and how this differs from a tactic, has been established, a solution can be designed to solve the problem of mimicking strategies, which shall need to encompass tactical and operational decisions as well.

Chapter 4 – AI Techniques applied to Fighting Games

4.1 Introduction

Having defined and contextualised strategies and tactics, both in martial arts and videogames, specific AI techniques can be explored that may be suitable to model strategies and tactics with a fighting game. Before proposing and designing a solution, a survey of current techniques must be carried out in an effort to understand the research that has been conducted, and ensure a novel approach is being adopted to solve the problem.

This chapter provides a literature review, which focuses on how AI techniques are applied in fighting games. The chapter begins by describing the role of AI in games across a variety of genres, helping to contextualise the proceeding sections. This is followed by a review of literature describing various game AI techniques and their commercial applications, as well as those that are focused on within academia. Specific game AI techniques such as finite state machines and various supervised and unsupervised machine learning techniques are discussed, providing insight into how each technique works. A detailed survey of the academic uses of these techniques is provided, including their usage in research related to fighting games. While literature on the application of AI within fighting games is limited, this chapter reviews various papers pertaining to this sub-field of research and evaluates the findings.

4.2 The role of AI in Videogames

John McCarthy, who first coined the term 'Artificial Intelligence' in 1956, defines AI as "the science and engineering of making intelligence machines, especially intelligence computer programs." (McCarthy, 2002) .The use of AI in videogames has been of increasing interest over the last few years, with its popularity rising due to the inclusion of AI techniques in videogames such as *Halo* from Microsoft (2001) and *F.E.A.R* from Vevendi (2005) (van Lent, 2007). van Lent (2007) describes the field of Game AI as a series of techniques to generate the behaviour of opponents, allies, or other NPC (Non-player characters). The purpose of implementing sound AI for NPCs in videogames is two-fold; first the NPCs add greater depth to the narrative of the game (where applicable), but arguably more important, NPCs provide the appropriate level of challenge to the player (Carneiro et al, 2014).

Laird (2001) states AI can cover a variety of aspects with videogames from providing strategic direction to groups of characters, generating commentary in sports videogames and dynamically altering the difficulty in single player videogames. It could be argued that the main driver behind AI in videogames is to allow for a more enjoyable experience by making for realistic non-player characters (NPCs). Fink

et al (2007) define NPCs as 'visible components' within videogames that are controlled by the computer itself, and can either work with or against the player. Khoo and Zubek (2002) concur in that NPCs can be used to either aid or hinder the player, and describe them as being capable of interacting with their environment as well as each other as AI has many applications in videogames.

A further motivation for using AI within commercial videogames is to centre the videogame further on the player's requirements for an enjoyable experience. AI can be used to observe the player's experience and modify certain videogame parameters to make for what may be perceived as a more fun play through of a videogame. This is the essence of player modelling and can be extended through various aspects of the videogame. Charles et al (2005) emphasised that commercial videogame development studios only invest a limited amount of time understanding their demographic and may not want to jeopardise a tried and tested design formula for the sake of what may be a minority. It is argued that a videogame can be designed to satisfy its core demographic, while also being flexible enough to cater for a wider user group. This is achieved by utilising adaptive player modelling techniques (Charles et al, 2005).

Buro and Furtak (2003) argue that the reputable area for AI usage is in board games such as chess and checkers, due to the turn based decision-making nature of these games. Buro and Furtak consider Real Time Strategy (RTS) games as not reaching the heights of board games with regard to AI usage, the same could be said for the majority of videogame types. Generally speaking, real time videogames, regardless of genre (fighting, action, sports etc.); all face the challenges in implementing AI. This is largely due to their fast nature when compared with turn based games, such as board games which are typically perfect information games, ideal for brute force AI techniques such as minimax decision trees and other search algorithms (Lucas and Kendall, 2006), (Santoso and Supriana, 2014). However, turn based games including chess, go and Othello have also been implemented using complex co-evolutionary techniques (Yao and Teo, 2006), (David et al, 2014)

Nareyek (2007) conveys that most research being carried out regarding AI in academia is focused on enhancing techniques that are already being used in anger in the commercial videogames industry. It is suggested that there are further gains to be discovered by shifting the focus of Game AI research to solve problems such as automated content generation. This could include art to be used for in-game environments, as well as story generation for videogame narratives. Ramirez and Bulitko (2014) take this further by implementing a novel AI experience manager called Player Specific Automated Storytelling (PAST). PAST generates multiple permutations of a story and selects one that is best suited to the player's style of play. In a similar vein to story generation, Lee et al (2014) propose a novel use

of AI in videogames by putting forward a novel means of leveraging machine learning to generate commentary in sports games.

4.3 Game AI Techniques

In this section, existing AI techniques that are used in commercial videogames, as well as videogame related academic research, are discussed. Techniques are presented with a variety of views from existing literature. The purpose of this section is to introduce the techniques and provide an overview for how they are used.

4.3.1 Finite State Machine

Johnson and Wiles (2001) discuss some of the common AI techniques used in modern videogames as well as their limitations, in particular, the concept of a finite state machine is described as being a set of states, a set of inputs, a set of outputs and a set of transition functions. The initial input and state are passed through the transition function and the new state, as well as the set of outputs, are calculated (Johnson and Wiles, 2001). van Lent (2007) offers some clarification on the workings of the finite state machine by putting it into the context of a basic videogame. A character may be in a particular state, such as an 'Attack' state, and as such, would act accordingly (in this case, by attacking). Once this characters' health drops below a certain threshold, the state would alter to a 'Flee' state, where the character would run away in search of health (van Lent, 2007). In this example, the aforementioned 'inputs' (Johnson and Wiles, 2001) would be the metrics to cause the state transition, such as a character's health. The outputs would be the actions performed whilst in a given state, such as attacking, and the transition functions would be the triggers that cause the transition from one state to another, such as dropping health below a certain threshold. FSMs are amongst the most common AI techniques used in commercial videogames, with usage found in the *Quake* franchise (id Software, 1996), *FIFA* franchise (Extended Play Productions, 1993) and *Warcraft* (Blizzard Entertainment, 1994) (Mallouk and Clua, 2006). According to Zhou et al (2006), typically, basic NPC emotion is handled by FSMs.

Houlette and Fu (2003) suggested that a finite state machine can be formally described as a set of states, S, a set of inputs I, and a transition function T(s, i) responsible for mapping a state from S and input from I to another state. The set S contains an initial state as well as zero or more accepting states. Once all information within the FSM has been processed, the ending state must be deemed as an accepting state to allow the machine to accept the input.

In practice, the FSM is a description of how an object can change its state over time in response to the environment and events that occur. Each state in the FSM represents a behaviour, resulting in behaviour changing as states change from one to another. The function T resides across all states, meaning that the states shall be left and entered in accordance to fulfilling the transition criteria for that particular state. The input is fed into the FSM continually as long as the game is active (Houlette and Fu, 2003).

Finite state machines have been used in commercial videogames such as ID Software's first person shooting games; Doom and Quake. The use of finite state machines in videogames is promoted by many developers due to their robust nature as they are easy to test and modify (Johnson and Wiles, 2001). However, the primary limitation of finite state machines lies in its predictability. The actions performed in a given state do not alter as time goes on, nor do the triggers that cause state transitions. This is to say that the entire finite state machine is a static, rule based system (Johnson and Wiles, 2001), rather than a system that is capable of learning and evolving as the game is played. Once a player has found a way to counter the finite state machine logic, they could exploit the static nature of the technique and use the same tactics to succeed each time. By the definition of AI put forth by Rich et al (2001), one may argue that finite state machines are not representative of a valid AI technique as they do not adapt or learn from their environment.

The static and predictable nature of hard-coded finite state machines becomes evident when FSMs are faced with situations not considered by the designer (Fairclough et al, 2001). This shortfall can be addressed to a degree, by implementing data driven finite state machines. The data driven approach uses authored data that structures the FSM. A data driven FSM is useful for instantiating custom FSMs whose states and transition logic are defined in an external file (Rosado, 2003).

Further to data driven finite state machines, other variants that are of interest include stack-based state machines and hierarchical finite state machines. The stack-based state machines, or push-down automata as they are sometimes called, use a stack-based data structure to store a memory of sorts (Houlette and Fu, 2003). The stack can be used to store relevant history, or even entire FSMs. The stack can also be used to record the sequence of states it has travelled through, enabling it to retrace its steps if necessary. The FSMs are stored as objects in stack, with the top most FSM being the active machine. Once the active FSM reaches its final state, it is popped from the stack and the next state in the stack is pushed up (this would be the parent of the recently popped FSM).

Additional variations of the traditional FSM were put forward by Mallouk and Clua (2006) in the form of Hierarchical State Machines, which are FSMs in their own right. However, each state can be broken

down into further FSMs. For example, the state 'Move' could contain sub-states 'Run' and 'Walk', which would have their own transition functions for exiting one sub-state and entering another. The hierarchical FSM makes use of its states by storing entire FSMs within them. When a transition to a particular state occurs, an entire further FSM can be invoked. This may be of particular use when dealing with strategies, tactics and operations. The 'child' FSM is treated as an entirely separate FSM, with its own data set being used to drive the transitions. In this instance, 'Move' is referred to as a composite state. Mallouk and Clua (2006) took the notion of Hierarchical State Machines a step further by applying concepts of Objet Oriented Programming. By applying principals of inheritance and abstraction, state machines can either be instantiated from concrete state machines (similar to concrete classes in OOP), or inherit from abstract state machines (similar to abstract classes in OOP). With this approach, the state machine is regarded as an OOP class, while the states themselves, both base states and sub states, are regarded as OOP methods. Mallouk and Clau (2006) ultimately regard Object Oriented Hierarchical State Machines as Finite State Machines that can be expressed via UML notation and can allow for large state machines to be created, leveraging existing state machines where possible to achieve this.

While research on machine learning appears to be the focus for game AI moving forward, there is still active research being carried out pertaining to the usage of traditional techniques such as finite state machines in videogames. Kenyon (2006) used finite state machines as part of a subsumption architecture, designed to house parallel levels of independent behaviours, working on a bottom-up basis.

4.3.2 Artificial Neural Networks

An alternative to the deterministic approach offered by FSMs, albeit a commercially less popular alternative, would be Artificial Neural Networks (ANN), which are an example of machine learning. Barto and Dietterich (2004) described a supervised learning algorithm as one which is fed a set of training examples as an input and produces a classifier as an output. Supervised learning relies on an algorithm that is provided examples of inputs and the outputs they should produce. This is referred to as training. Constants within the algorithm are adjusted to support the training examples. Once the training has been completed, the algorithm can be put into practice and generate outputs from real data that is fed as inputs.

Biological neurons are the recipients of stimulus signals passed on by other neurons, and once a threshold activation level is reached, the neuron fires signals to all other neurons that are connected (Cant et al, 2002). ANNs are inspired by neurons in the biological brain. ANNs are used to allow the AI

system to be updated with knowledge as the game state alters whilst the human player progresses to play the game (Johnson and Wiles, 2001). Rather than the static rule based approach used by finite state machines, the use of Neural Networks provide a system in which the AI agent can learn the players' tactics and act accordingly. This forces a player who constantly uses the same technique to alter their style as the AI will now adapt to the style that is constantly repeated (Johnson and Wiles, 2001).

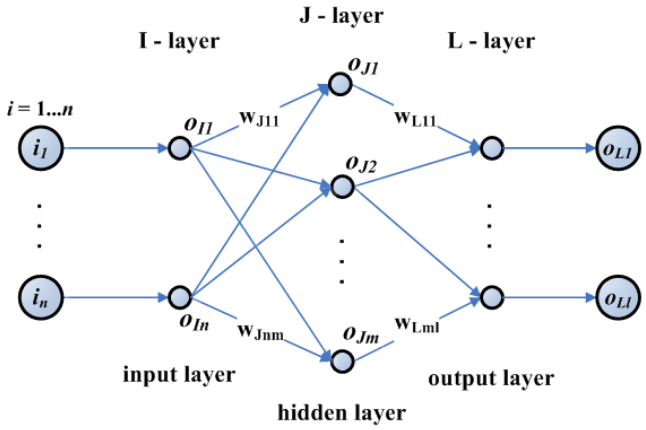

Figure 4.1 – A Multilayer Artificial Neural Network

Figure 4.1 shows a multi-layered perceptron (University Of Maribor, n.d.), which is a type of ANN consisting of an input layer of neurons, the *i* layer, which are passive and contain information on inputs passed to the network, to which it must respond. The *l* layer of neurons represent the outputs and correspond to actions that can be carried out once the ANN computes the response to the inputs. Between the input and output layers, there can be a number of hidden layers, used to facilitate the response of the network (Chaperot, 2006). This is represented in Figure 4.1 as the *j* layer of neurons.

The learning phase of a Neural Network training is nothing more than adjusting weights, *w*, between nodes, *o*, (often referred to as 'Neurons') such that a desired output value is achieved. Input values containing metrics that describe the current state of the game are each multiplied by a weight value that is initially assigned a random value, prior to entering a 'hidden' layer (see Figure 4.1). The randomly assigned weights are optimised during training and then fixed. Within the hidden layer, a calculation is made, in which the weighted sums of the input nodes are determined. The function used for the calculation here is usually the sigmoid function as it allows networks to be trained using techniques such as backpropagation. A similar calculation is performed between the hidden nodes

and the output nodes in order to obtain the output values (Coppin, 2004). Usually, a value is calculated for each output node, and the node with the largest value is 'fired'. This is to say that if a given node has the largest value assigned to it, a particular action shall be performed.

As the initial outputs are based on random weightings, it is highly unlikely that the node to be fired would be the desired output. This is why, prior to the initial run through of the neural network, training data is provided. The training data contains rows of data, each with a set of inputs, and the desired output values corresponding to the inputs. Once a neural network has been 'trained' it should be capable of providing the desired outputs for inputs similar to those found in the training data (Coppin, 2004). Having completed an iteration of calculations based on the initially random weights, a neural network shall manipulate the weights along each connection such that the desired output (or a value close to it) is achieved. This is done by using an algorithm called back-propagation. The back-propagation algorithm has been the focal point of much research to date. In particular, Cho et al (2006) use this technique to create an AI player within a fighting game. The back-propagation algorithm involves initialising all weights to random values and passing a given input through the ANN. Following this, outputs are generated and differences between the current outputs and desired outputs are calculated. Deltas are also calculated for the hidden layers and the weights are redistributed, following which input patterns are passed through the ANN again and these steps are repeated until the weights are distributed such that the desired outputs are attained (Chaperot, 2006).

This particular use of neural networks is called 'supervised learning' as the desired output is known for a set of given inputs in the training set. Neural networks have been implemented in several modern videogames including Battlecruiser 3000AD, Heavy Gear and Dirt Track Racing (Johnson and Wiles, 2001). A popular example of a successful implementation of an ANN was Anaconda, an AI designed for chess games using 5046 weights (although no details were provided on the internal network structure). The inputs to the network are the current board positions, with the output being a value used in a mini-max search (Lucas and Kendall, 2006).

Chaperot, et al (2005) launched an investigation into the use of ANNs in a motorbike racing game. Two Neural Network techniques were used, Backpropagation and Evolutionary algorithms. The aim of the research was to implement an ANNs in the Motocross racing game such that the bike is ridden as well as possible, but while still retaining the characteristics of a human playing the game. The premise of the ANN is simply to input game state data including the position of the bike in way point space, front and right directions of the bike in way point space, the velocity of the bike in way point space, height of the ground, and position of track centre lane (Chaperot et al, 2005). The outputs include controls

such as accelerate, brake, turn left/right or lean forward/back. As well as using backpropagation, Chaperot, et al (2005) also used Evolutionary algorithms to train ANNs. This involves adjusting the weights of the ANN by using the genetic algorithm first established by Holland (1984).

Chaperot's (2005) research was focused on AI using ANNs in a racing game, however, it would appear that there is very little research conducted with regards to the use of ANNs in one-on-one fighting games. Cho (2006) used the backpropagation algorithm, only this time it was used in a fighting game. Cho's (2006) laid out the basic rules of play for the game, where it is stated that the IC (intelligent character) can either be idle, perform one of five attack moves, each with varying effects, move or guard. Each action takes a number of 'clocks' as a measure of the time taken to perform the given action. The actions must each be performed within a specific distance interval. As inputs, the ANN takes the opponent character's (OC) current action, the step of this current action (for example, if the current action takes 4 clocks, the current clock of the action), the relative distance between the fighters and the OC's past actions. The outputs of the neural network encompass the aforementioned actions that can be carried out by the intelligent character.

The backpropagation neural network gave better results in terms of time, however, Chaperot et al (2005) stated that the neural network did not deal well with unusual circumstances such as recovering from a crash, or facing the wrong direction. This limitation is most likely due to the fact that the training data does not adequately account for these eventualities. It is stated that the genetic algorithm neural network is better at adapting to new situations than the backpropagation neural network (Chaperot et al, 2005). As such, the research conducted by Chaperot, et al (2005) limits the game to a single, albeit 'long' track. A measure of success of the neural network would be to see how it performs on different tracks. Humans can adapt to different tracks quite well as the basic principles are the same, such as slowing down when taking a sharp bend, or speeding up along a straight portion of the track. There is no evidence in the research being presented by Chaperot, et al (2005) that this is the case for the ANN they trained using the backpropogation algorithm.

Based on the work of Chaperot et al (2005), it could be argued that ANNs are better suited to racing games, rather than fighting games, which are far more complex. An experiment was conducted by Chaperot et al (2005), where a bike had to race on a single long track, with many obstacles that needed to be negotiated. The track was completed by a good human player, an ANN trained using back propagation and an ANN trained using a genetic algorithm. The results yielded by the experiment showed that the backpropagation neural network gave better performance in terms of time taken to complete the track than the genetic algorithm neural network (Chaperot et al, 2005). It is not divulged

in the paper whether or not the AI outperformed the human player. This experiment can be commended for its inclusion of a human player to gather data.

4.3.3 Bayesian Belief Networks and Naïve Bayes Classifier

A further supervised learning technique is the Bayesian Belief Network. Bayesian Belief Networks are probabilistic graphical models that represent attributes and the dependencies between them, and can be used for classification. Hyde (2008), who describes how BBNs can be applied to the game Thief (Looking Glass Studios, 1998), states that BBNs have a graphical component, referred to as a directed acyclic graph (DAG), the nodes of which represent random variables, and a numeric component. Parent / child relationships are illustrated on BBNs between nodes, with an arrow pointing from the parent node to the child node. This illustrates a dependency between the two variables, and it is from this point that a conditional probability table (CPT) is created for a given variable which contains the probability of that variable, given the probabilities of its parent variables. The domain of each variable contribute to the complexity of the network, for example, for Boolean variables, the complexity shall be lower as there are only two possible values for the variable. The structure of the graph largely depends on the context of the data being classified. By creating the DAG and then populating the CPT, the probabilities of variables, or events occurring in a game, can be calculated. In He et al (2008a), BBNs are used to classify data using the DAG and CPT. The Naïve Bayes Classifier (NBC) is a much simpler approach to classification which simplifies the problem by assuming attributes are independent of the target value. The problem typically involves a set of training data, then a new instance of the classifier produces a target value using equation (1):

$$v_{NB} = \underset{v_j \in V}{\arg\max}\, P(v_j) \prod_i P(a_i \mid v_j) \qquad (1)$$

Where $\mathbf{V}_{NB}$ is the class value output by the classifier, and a_i are the values for attributes fed into the classifier. $\mathbf{V}_j$ denotes elements of the set V which are the possible target / class values. In He et al (2008a) where NBC and BBN are applied to the game of Pac-Man, the NBC is said to be typically less accurate than BBN due to its ignorance, however, it is computationally quicker (He et al, 2008a).

4.3.4 Reinforcement Learning

Barto and Dietterich (2004) describe reinforcement learning as being applicable to problems where specific examples of desired inputs and outputs are not available, but some criterion for good and bad behaviour has been identified. This falls in line with the definition put forward by Danzi et al (2003),

who suggest the reinforcement learning involves an AI agent acquiring intelligence only through interactions with its environment. Arguably, reinforcement learning falls into the category of supervised learning. However, it differs from supervised learning in that it does not use examples of desired performance, but only sends a reward signal to suggest whether a particular input/behaviour yielded good or bad results.

Barto and Dietterich (2004) used the analogy of trying to acquire stronger reception on a mobile phone. Individuals tend to walk around aimlessly, asking the person on the other end of the phone whether the signal has improved. They do not specifically know where to walk, however they can ascertain whether or not their actions have yielded good or bad results by the response from the person on the other end of the line. This can be expressed as a problem where *R* represents the reward function for various geographies *x.* We say that *R(x)* is the signal strength that can be obtained at point *x*. The reinforcement learning problem is to identify location *x** such that *R(x*)* yields the greatest signal strength.

In Danzi et al (2003) where reinforcement learning is used to control the difficulty of a fighting game, a reinforcement learning framework is summarised as having an agent, the current state of which is denoted by state, $s \in S$, which chooses an action, $a \in A$, leading to a new state, s'. As also stated by Barto and Dietterich (2004), the reward signal, R(s, a), which is fed back to the agent each time action *a* is executed in state *s*. The main aim is articulated as being; to maximise the return, representing the expected value of the future rewards. For example, one could consider reward as being a monetary value, in which case, going to school and acquiring a good education would be actions that maximise the expected reward. This is achieved by learning an optimal policy $\boldsymbol{\pi}^*$ which maximizes the expected return by mapping state perceptions to actions. The optimal policy can be constructed if the agent learns the optimal action value function *Q*(s,a)*, where for each state *s*, the best action, *a* is that which returns the maximum value for *Q*. There are a variety of learning algorithms in existence to solve problems such as this, including the Q Learning algorithm (Barto and Dietterich, 2004). Batalov and Oommen (2001) argued that many algorithms do indeed exist to solve reinforcement learning problems, however, the majority make an underlying assumption that the environment is fully observable, which in most cases proves to be a false assumption. The Q Learning algorithm is defined by equation (2):

$$Q(s,a) \leftarrow Q(s,a) + \alpha\,[r + \gamma\,.V(s') - Q(s,a)] \qquad (2)$$

Where V(s') = $\max_a Q(s',a)$, α is the learning rate and γ is a discount factor (Danzi et al, 2003). The discount factor determines the relative value of deferred versus immediate rewards (Batalov and

Oommen, 2001). Graepel et al (2004) went on to say that reinforcement learning is ultimately based on the Markov Decision Process (MDP), and is characterised by the tuple (S, A, R, T) where S is the set of states; A is the set of actions; R is the average reward function responsible for assigning reward to the agent after an action is performed moving from s to s'; and T denotes an unknown stochastic, transition dynamics, which gives the probability of a transition from state s to s' if action a is performed.

4.3.5 Clustering

Clustering, which is considered to be unsupervised learning differs from supervised learning in that training data are not available. While pre-defined classes are known at the time for supervised learning techniques, unsupervised learning does not assign inputs to known outputs based on predefined classes. Instead, unsupervised learning 'clusters' data based on inputs. Keller (n.d.) first described the distinction between classification and clustering as hinging on the need for predefined classes. Classification requires these predefined classes before data can be classified; however, clustering relies on learning classification from the data itself.

Anagnostou and Maragoudakis (2009) defined clustering as a data mining process that differs from classification in that the clusters are not predefined in that the datasets are not labelled beforehand and belong to a known class. Clustering is defined as an unsupervised learning technique, for which the main criteria are defined by Anagnostou and Maragoudakis (2009) as follows:

(i) 'Data contained within a cluster present similar characteristics.'

(ii) 'The distance between each point of a cluster and any other point within a cluster is smaller than the distance between each point of a cluster and any point of a different cluster.'

Anagnostou and Maragoudakis (2009) use clustering to define player types, assigning them to one of many clusters based on how they play a Space Invaders type game.

Hierarchical clustering is an unsupervised learning technique used in data mining to group like elements into sets, or clusters, without any prior knowledge of the data classification. To cluster data, a metric and linkage criterion must be set. The metric used in this research is the Euclidean distance, which will serve as a measure between elements. The linkage criterion serves as the measure between clusters. In this research, the complete linkage criterion is used as defined in equation (3):

$$D(X_i, X_j) = \max_{x \in X_i} \max_{y \in X_j} d(x, y) \quad (3)$$

Where $d(x, y)$ is the distance between elements $x \in X$, and $y \in Y$, and where X and Y are two sets of elements. The distance between two clusters is defined as being the distance between the two furthest elements (Fernandez and Gomez, 2008).

4.3.6 Evolutionary Algorithms and Genetic Algorithms

Evolutionary computation is a broad field of research that encompasses all computing solutions based on biological evolution (Di Pietro et al, 2006). Genetic algorithms are a search heuristic that create candidates through evolution, and can be used in scenarios where little to no information is available regarding the problem. They are particularly useful where the optimal solution is not necessarily required (Cho et al, 2007). Johnson and Wiles (2001) described evolutionary algorithms as search procedures that reflect concepts of natural selection and natural genetics. Genetic algorithms are a subset of evolutionary algorithms commonly used in videogames.

Genetic algorithms work amongst an initial population, containing randomly generated states (also referred to as candidates). These states each have a string (also known as the chromosome) and fitness value associated with them. Based on the fitness value (which is determined by the fitness function), the states create offspring by randomly cutting off the string of one state. This string is then merged with the remaining string of a further state with which it is creating offspring. This process is called crossover, and can be followed by mutation which allows for the string to be tweaked slightly based on random probability (Di Pietro et al, 2006) (Belew et al, 1991). This concludes the first generation of evolution. It is not uncommon to have thousands of generations to solve a given problem.

4.4 Application of Game AI Techniques

AI has played an increasingly important role in modern videogames. The techniques used in commercial videogames are still limited. As discussed earlier, finite state machines are too static to provide a varying challenge, and machine learning techniques have only been applied to modern videogames in a limited manner (Johnson and Wiles, 2001). In order to gain a greater appreciation for the potential usage of AI techniques in videogames, more specific domains of research must be surveyed. This section reviews the usage of the previously discussed AI techniques in terms of how they are used together in the context of videogames.

4.4.1 Player Modelling

Player modelling is a learning technique described as being lightweight, simple and flexible enough to be implemented in otherwise computationally expensive videogames (Houlette, 2003). At a high level, player modelling involves the game AI maintaining a profile of each player, including their preferences, weaknesses, strengths and other characteristics. The profile, or model, is updated as the player interacts with the game environment and other non-player characters. Player modelling allows the computer player to evolve with time as the profile can be updated over several play sessions (Houlette, 2003).

The player model itself is essentially a collection of traits that are used to track the players' use and frequency of various actions. For example, in a first person shooting game, the trait 'Uses Grenades' would track information regarding the players' use of grenades (Houlette, 2003). The traits must be assessed on the basis of a pre-defined semantic. For example, the *Uses Grenades* trait would be useful when assessed against the 'knowledge and proficiency' semantic, as this would record data related to how well the player uses grenades. Another semantic which may be useful would be the 'habits and preferences' semantic. This would tell us how often the player uses grenades, but would not include information on how effectively they are used. Traits within the player model can be as broad or as refined as the designer wishes. Naturally, there is a trade-off for more detailed models as they take longer to design and are computationally more expensive than their broader counterparts. The rather broad trait, 'Uses Grenade' could be refined to several traits such as 'uses grenade while retreating', 'uses grenade while covering' etc.

When designing the player models it is important to ensure they are tightly integrated to the game AI, as the player model describes the traits of a given player, but the AI must make use of this data. It is important that the traits and semantics complement those that are typically used in the game (Houlette, 2003).

An alternative design approach to player modelling is the use of hierarchical player models. Rather than using a flat list of traits and their associated values, the traits (referred to in this scenario as 'concrete traits') can be categorized into high level abstract traits (Houlette, 2003).

A basic player model can be implemented by simplifying the problem. The player model can be viewed as a statistical record of the frequency of some interesting subset of player actions (Houlette, 2003). By considering this definition, a player modelling system must provide a mechanism through which player statistics can be observed and updated accordingly. Further to this, the system must also provide an interface through which the model can be queried for player information. Houlette (2003)

implemented a basic player modelling class using C++. The model is based on the 'Habits and preferences' semantic, with the traits, represented here as floating point values between 0 and 1, being stored in an array. The float value of the trait reflects how frequently it is used. The method for updating the model entails updating the trait values. This is achieved by using the Least Mean Squares training rule:

$$traitValue = \alpha \,.\, observedValue + (1 - \alpha).\, traitValue \qquad (4)$$

The update to the player model is enacted in two phases. Firstly, the game AI must know when to update the model. This is typically when the player performs an action that would alter the statistical record being maintained regarding the players' actions. This is typically hard coded into the videogame itself and can be computationally expensive. An alternative to trait detection is to implement Finite State Machines. The second aspect of the update, is calling the update method itself to alter the player model in accordance to the action that has just been carried out by the player. The FSM approach can be used for both trait detections and updating the model itself. This approach decouples the player modelling system from the videogame code, making for a cleaner, repeatable and modular system.

Various classifiers can be used for clustering traits to certain player types within Player Modelling. Player models are little more than a collection of statistics indicating the preferences of the player, allowing certain player types to be categorized and labelled. Current literature uses a variety of classifiers such as Bayesian Belief Networks, Naïve Bayes, K Nearest Neighbour, Radial Basis Function and ANNs. He et al (2008b) describes an RBF network as being a type of ANN, but one that uses radial basis function as the activation function. In this instance, the RBF network yields stronger results than feed forward ANNs trained with backpropagation.

A clustering approach to player modelling is presented by Anagnostou et al (2009) using a space invaders variant as a test bed. The game presents the player with the typical space invaders scenario, but offers a choice of four weapons, the ability to 'energize' asteroids such that they are jettisoned into enemy ships, and a variety of enemy types. The additional game features allow for two high level player strategies; the first is to use an all-out brute force attack, which can be time consuming; or the second which involves skilfully energizing asteroids such that they damage enemy ships. The latter is less time consuming but requires more skill to achieve. Anagnostou et al dubbed those who employed these styles as 'action player' or 'tactical player' respectively. Data were collected from 10 students who played the game four times each, making for 40 sets of the 19 attributes that are collected during each play. The data are clustered using the 'Clustering Using Representatives' algorithm (CURE), which is a clustering technique that leverages both partitioning and hierarchical clustering (Anagnostou et

al, 2009). The algorithm begins with a fixed number of points, *c*, being selected from within each cluster. The points are selected such that they are relatively far apart within their own cluster, as well as being far from the centre of the cluster, providing a representative set of points for a given cluster. These points are then shrunk to approach the centre of their respective cluster. It is at this stage that the hierarchical clustering is used to merge clusters where representative points across different clusters are relatively close. Upon merging clusters, new representative points are selected and shrink towards the centre once again. The CURE algorithm successfully classified two of the game features synonymous to the two player types.

He et al (2008a) used Pac-man as a test bed to investigate the use of Bayesian Belief Networks and Naïve Bayes classifiers for player modelling. Pac-man is described by Gallagher and Ledwich (2007) as a predator-prey game where the human player manoeuvres Pac-man through a maze while he is being pursued by ghosts. Pac-man scores points by consuming dots and while avoiding contact with the ghosts. He et al (2008a) took a similar approach to He et al (2008b), by first collecting game metrics across 12 attributes. Noise modelling and attribute sub selection are also incorporated into the methodology as demonstrated by He et al (2008b). Bayesian classifiers are used to statistically predict class membership probabilities, such as the probability that a given dataset belongs to a particular class. This is different to clustering as defined by Anagnostou et al (2009), which is an unsupervised learning approach for categorizing data without knowing what each class represents.

Hyde (2008) presented a further use for BBN within the *Thief* (Looking Glass Studios, 1998) videogame, modelling the variables relating to a guard's decision to act. In the example, the scenario involves a videogame where the player assumes the role of a thief that must bypass a guard. The thief may make movement, noise and leave footprints. However, rats are another factor of the game world that make noise and movement, but do not leave footprints. The BBN shows two nodes, Rats and Thief at the parent level. These nodes have child nodes, of which there are three in total. The Rats node has two child nodes; noise and movement. The Thief's node has three child nodes - it shares the noise and movement nodes with the Rat parent node, but also has a footprints node. Notice that there is a relationship of cause and effect, with the top level nodes acting as the cause, whereas the lower level nodes act as the affect.

Each variable in the above example has a Boolean value, true or false. The BBN works by using the probabilities assigned to each variable to determine what the outcome of a given event should be. For example, given that rats are moving and a thief is moving, depending on the probabilities assigned to the variables, we could determine the percentage chance that the guard may notice movement. When

assigning probabilities it is best to begin with parentless nodes, and then the child nodes. For the child nodes, the probabilities of the corresponding parent nodes must be factored in. For example, for the movement node, we must assign probabilities to movement given that the rats and the thief are moving; only rats are moving; only the thief is moving; and finally, nobody is moving. The BBN can be used to feed into rules regarding the behaviour of the guard, and the model can be updated if necessary, reassigning probabilities.

The example outlined by Hyde (2008) is a direct use of BBNs to control the actions of a non-player character, however, BBNs can be used to classify data in a similar way. Given the values of certain attributes, probabilities can be assigned to possible values of the attributes. Given the correct probability distribution, it can be deciphered that a player is of a certain type, meaning they conform to a given player model.

Data are collected for three identified strategies as in He et al (2008b). The results of feeding the data into the Naïve Bayes and Bayesian Belief Network classifiers show that both classifiers perform well, correctly classifying approximately 85% of the data. Two sets of experiments are conducted, one with noise modelling and one without noise modelling. The experiment without noise modelling shows the Naïve Bayes classifier slightly outperforming the BBN classifier. In this experiment, the attribute sub-selection stage makes negligible performance enhancements. The experiment with noise modelling shows a significant performance increase in terms of accuracy, with each classifier scoring in the 90th percentile for successfully classifying the data.

Aspects of player modelling are explored by Charles et al (2005), particularly its application as a means to make videogames more accessible for a variety of different players. Charles et al suggested an approach whereby the player is monitored in real-time (while the game is being played). The player is monitored and data representing player preferences are collected to feed into a model and the game is adapted to conform to this model. It could be argued that a similar approach is adopted by Drachen et al (2009) in their study of self-organizing maps and player modelling.

Drachen et al (2009) have explored player modelling in the commercial game Tomb Raider Underworld (Crystal Dynamics, 2008), using self-organizing maps, which is a form of clustering. The self-organizing maps are trained on the playing behaviours of 1365 players that completed the game. Relating to the context in which player modelling is discussed by Charles et al (2005), Drachen et al has cited the replacement of traditional play testing to ensure the game is being played as it was meant by the designers as a motivation for the automated approach. The approach can also assist in the tailoring of the game to conform to a player's style. Game metrics (detailed numerical data relating to how the

game is played) are collected via a game metrics logging system, Eidos Metric Suite, developed by Eidos, the publisher of Tomb Raider Underworld (TRU).

There are a total of six extracted features from TRU, some of which include cause of death, total number of deaths, completion time and help on demand. The game metrics collected from these six overarching features are fed into a large scale self-organizing map, called an emergent self-organizing map (ESOM).

Means of extracting game data from commercial games are discussed by Hsieh and Sun (2008), using developer provided tools to collate player statistics. The highest performing ESOM revealed four player types; runners, veterans, solvers and pacifists. Drachen et al (2009) argued that the number of clusters identified is significant given the nature of the game as TRU is a relatively linear adventure game. The research conducted delivers a promising approach to player modelling at the industrial standard. It would, however, be interesting to see the effects of this approach on a less linear game that offers more control over the player's actions, opposed to the environment. TRU allows the player to perform several actions and interact with the environment in a variety of ways, but offers little in terms of deviating from the standard approach of play and does not allow the player to express themselves using their own strategies. A fighting game has a greater focus on controlling the character in a variety of ways, providing an intricate gameplay system allowing the player to control each of the limbs of the on screen fighters. In games such as this, there is typically less focus on the environment / game world, and more control on the protagonists.

It is difficult to conduct research on popular commercial games, however, Drachen et al (2009) succeeded in conducting research at the commercial level using industrial level tools and should be commended for shedding light on what is otherwise an incredibly locked down area of game AI research. Another commercial game that has been studied in the context of player modelling is *Super Mario Bros* (Nintendo, 1985). Pedersen et al (2009) attempted to model player experience in the public videogame Infinite Mario Bros, a clone of the classic Nintendo game, Super Mario Bros. The research conducted by Pedersen et al is more concerned with modelling player experience rather than directly modelling player strategies derived from gameplay interactions. The game is a classic platform game set on a two-dimensional plane. The player controls the Mario character on screen and can move left or right, jump (gravity acts on Mario) and shoot fireballs.

Pedersen et al collected data from 'hundreds' of players from the internet, and then categorized the data into one of the three following categories: (i) Controllable Features, (ii) Gameplay Characteristics, and (iii) Player Experience. Infinite Mario Bros has one major feature that sets it apart from the

Nintendo original that it is cloning; the ability to generate levels. The first data classification, controllable features, relates to this feature. The data within this category relates to players' choice regarding the level design and is of little interest in the context of strategy player modelling. Category (ii) data relate to the game metrics such as jump frequency and context of the jumps. Data in category (iii) relates to an online survey completed by players regarding their emotional response to a given level. This is of little concern given the context of the research being conducted in this Thesis.

Pedersen et al used an ANN (single layered perceptron) to establish a relationship between data captured in category (ii) and data captured in category (iii). It was decided that attribute sub-selection must be performed on the data. Rather than using a data mining tool such as WEKA, as used by He et al (2008b), Pedersen et al compared three schemes - nBest, Sequential forward Selection and Perceptron Feature Selection. It was determined that neither of the three approaches were likely to provide an optimal feature set as neither searches all possible combinations. Pedersen et al concluded that by using non-linear preference modelling, given a set of game metrics (inputs), the ANNs can be trained to predict the emotional response of the player.

4.4.2 Pattern Recognition

Pattern recognition can be used as an integral part of game AI. It can form the basis of an AI method by feeding into a decision making system. It is common to split an AI system into two components; pattern recognition and the decision making component. Pattern recognition is used to extract relevant information from data. The most basic form of pattern recognition is pattern matching (Kaukoranta et al, 2003-1). This involves little more than searching for a predetermined pattern from a given input, such as matching a word amongst several characters of text. In pattern recognition, we refer to a measurement as a symbol, which can be a quantity, label or even a combination of primitives. We refer to a set of features as an alphabet (Kaukoranta et al, 2003-1).

There are two common ways in which pattern recognition can be utilised; Prediction and Production. We consider a set of features/symbols, S, which is a subset of the alphabet, Σ (Kaukoranta et al, 2003-1). The set, S, denotes the symbol sequence. In the context of a videogame, this is the sequence in which features are presented in the game world. This could be a sequence of actions, such as punches and kicks in a fighting game. The sequence, S, is modelled by considering the dependencies between features. If we consider the game world as being formed of events and states, then we can use 'generators' to label the events and states with symbols (Kaukoranta et al, 2003-1). For example, a move performed by a character in a boxing game could be labelled by the generator as being 'jab',

'uppercut', 'cross' and 'hook'. Once a symbol sequence has been generated using the generator, a model can be constructed.

The model for pattern recognition can be constructed by recognizing the dependencies between symbols. These dependencies are typically stronger with symbols that are close to each other. The model is queried by the decision making system and used for either *prediction* or *production* (Kaukoranta et al, 2003-2). When predicting the next symbol, the pattern recognition mechanism passes the observation to the decision making system, possibly in the form of a probability distribution of the occurred symbols, rather than a particular symbol. This approach can be used to predict outcomes of a player to the AI's advantage, such as predicting the probabilities of the human player's next move and preparing an appropriate counter. The approach can also be used to produce action sequences, such as a chain of moves, mimicking a human player. It is the latter of these uses that pertains to the focus of this Thesis. The generator model is used to produce symbols. This is referred to as pattern generation rather than pattern recognition and can be used to model the sequence of actions of player (Kaukoranta et al, 2003-2).

Kaukoranta et al (2003-2) explore methods of implementing pattern recognition using various soft computing techniques. With regards to use of pattern recognition at the various levels of play patterns, Kaukoranta et al (2003-2) have recommended using supervised or unsupervised learning at the strategic level, largely due to its computational demands. This is to say that learning at the strategic level is infrequent, and in some instances can be completed ahead of runtime, so it is feasible to use a computationally expensive technique such as an ANN as a means of pattern recognition. The tactical level is faster paced and involves potentially complicated patterns, making for less thorough results. It is at this level that it is recommended to use hidden Markov models.

Pattern recognition is explored in game AI research and is treated as a component of the overall AI system. He et al (2008c) explored pattern recognition to identify a player's strategy in a 'Predator / Prey' genre game, with the intention of the results being fed into a decision making system that is implemented using upper confidence bounds. He et al (2008c) describes player strategy pattern recognition, as using a record to maintain each player's profile with details on skills, weaknesses, preferences etc. This method falls more in line with player modelling as described by Houlette (2003).

He et al (2008c) used the videogame of Dead End as a test bed for their approach to player strategy pattern recognition. Dead End is a 'Predator and Prey' game taking place on a 20 x 20 grid involving a cat (the player), two dogs (enemies NPCs) and an exit (the goal). The idea behind the game is for the cat to reach the exit, avoiding the two dogs, within a certain number of moves. The approach used by

He et al (2008c) was first to identify attributes that are of use for the data collection phase. In the context of Dead End, 12 attributes are chosen, including distances between characters, angles between characters and the distance between characters and the exit. The next phase, having selected the attributes that are required, involves collecting data for each of the attributes. He et al (2008c) identified three strategies and collected player data from each strategy 58 times, creating174 datasets across the three strategies. No explanation was offered as to why the game was played 58 times. Furthermore, it is important to note that the game is being played by simulated human players, not human players themselves. He et al (2008c) offered no detail on what method is employed to simulate the players.

The next step, once the sample data has been collected, is to remove certain attributes from the overall dataset. This is done to prevent the unnecessary attributes being included to the point that they have a negative impact on the useful data. To achieve this, He et al (2008c) used a data mining tool to automate the de-selection process, leaving them with seven useful attributes, opposed to the 12 that were initially selected. This reduced set of sample data is fed into a supervised learning algorithm, K nearest neighbour (KNN). Results of an experiment suggest that the KNN learning algorithm correctly classified approximately 92% of the strategies played during a game of Dead End. This is true for both the initial set of data (12 attributes) and also the reduced set of data following attribute subset selection (7 attributes). The results of the same data passed through a BBN correctly classified 93% of the strategies played using the 12 attribute set, and 100% of the strategies played using the 7 attribute set. However, there is no description of how the experiment was set up.

He et al (2008c) have shown that Bayesian classifiers are more effective than the KNN approach on sets of data that have been stripped down via an automated attribute subset selection process. However, with each of the three strategies being played 58 times for the purpose of data collection, one would expect to see results of this calibre. As there is so much training data, and only 3 strategies to choose from, the KNN or Bayesian network has a one-in-three chance of correctly identifying the strategy being played out. The percentage of success is provided, however, it would be more interesting to examine how many games were played, and with which strategy. A more rigorous test would be to use an alternative videogame, or modify Dead End such that it allows for more strategies, giving the player more freedom to create strategies. Results could then be used to decipher how useful this approach would be on a larger scale, or perhaps in a commercial videogame.

He et al (2008b) also explored the use of pattern recognition using the radial basis function (RBF). This research was conducted using the classic Namco videogame, *Pac-Man* (Namco, 1980) as a test bed.

The methodology followed was similar to that used by He et al (2008a, 2008c) and involved the same initial steps. These steps include identifying attributes for a complete dataset, collecting data from the game relating to these attributes and using a data mining tool to identify the most useful subset of attributes. Initially, 12 attributes were selected to represent the strategy being used by the player. The implementation of Pac-Man used was a simplified version of the original videogame. The Java implementation used here only consisted of one 'life' for the player and a single maze.

Data is collected from three possible strategies using 12 human players. Each player played the game using each strategy 20 times, making 720 sets of data. Noise reduction was used to nullify the impact any non-strategic data may have had on training the classifier. This is achieved by allocating any data that does not conform to one of the three identified strategies into a fourth strategy for noise. The noise strategy is used to train the classifier to identify when a human is not using one of the pre-determined strategies.

The WEKA data mining tool was used to perform data mining to find an appropriate subset of attributes from the initial 12. He et al (2008b) performed this on two different sets of data – the first with noise reduction, the second without. In each case, the attribute sub-selection process reduces the number of attributes to eight. However, each dataset provides a different attribute subset. It is noted that the attribute subset selection reduces the classifier's learning time (due to the reduction in data) and improves performance (due to less negative impact).

Two experiments were conducted, one on the set of data with noise modelling, and one on the set of data without noise modelling. To gain a context of the usefulness of the Radial Basis Function as a classifier, the results are compared to those of other classifiers including Support Vector Machines, Naïve Bayes and Bayesian Belief networks. The results for the dataset without noise modelling show that the Radial Basis Function (RBF) performs on a par with Naive Bayes (NB) and Bayesian Belief Networks (BBN), each classifier correctly identifying strategies approximately 85% of the time. The Support Vector Machine (SVM) performed poorly, only correctly identifying the strategies approximately 30% of the time. The results for the dataset with noise modelling are slightly stronger on most counts. The RBF performs stronger than all other classifiers, achieving an approximate 95% success rate in identifying strategies (after attribute sub selection). All other classifiers achieve similar results to their 'without noise' counterparts. Results provided include before and after attribute sub selection. While the results for attribute sub selection are better than those without attribute sub selection, the difference is negligible.

He el al (2008b) have demonstrated that the RBF is a powerful classifier for pattern recognition, at least on par with BBNs and NB networks. The importance of noise modelling is also demonstrated in He et al (2008b). However, much like the research carried out in He et al (2008c), where Dead End was used as a test bed for the KNN classifier, the strategies appear to be incredibly limited. Much like the aforementioned research, He el al limit their experiment to only three strategies and, while it achieves solid results, there is a large volume of test data that must first be accumulated before the comparatively small number of strategies can be correctly identified.

He et al (2008a, 2008b, 2008c) appear to have conducted very similar research across the fields of pattern recognition and player modelling. The experiments described in these papers are largely based on the same principles and appear to follow the same methodology each time; namely selecting attributes, modelling noise, using data mining to find the appropriate subset of useful attributes and then training a classifier of some sort to classify further instances. Player modelling and pattern recognition are closely related in that pattern recognition can be used to create a player model. The experiments conducted by He et al (2008a, 2008b, 2008c) fall more in line with player modelling. Essentially, He et al are training a classifier to identify a player's preferences based on the data provided. If the data are from a player who uses a strategic approach, rather than an all-out action based approach, these data are recorded and fed into the classifier which then classifies the player appropriately.

4.4.3 Mimicking

While not directly related to gaming, mimicking is a key factor of the research carried out in this thesis, and much literature related to mimicking pertains to AI and robotics. As suggested by the term, mimicking in the context of artificial intelligence and robotics focuses on copying the actions of a given subject. Extensive research has been carried out with regard to mimicking, much of which draws inspiration from nature to extract practical ideas that would be beneficial to mankind if it were possible to mimic them using robots (Bar-Cohen, 2012). Robots are defined as being electromechanical devices that retain biometric properties, giving them great advantages in performing complex tasks and operating in conditions that are deemed out of reach or too dangerous for humans (Bar-Cohen, 2012).

Bar-Cohen (2012) describes various applications for mimicking nature using a variety of techniques. Examples include how fishing nets mimic spider webs and how aircraft structures draw inspiration from honeycomb structures. Bar-Cohen (2012) regards mimicking humans to be a great challenge and mentions how AI plays a role in achieving this, sadly details on specific techniques are not provided.

Robots that are designed to mimic humans are referred to as humanoids, and mimic the general appearance of humans in terms of having arms, legs and a head (Bar-Cohen, 2012).

Research has been carried out in field of mimicking for humanoid robots. Due to the magnitude and complexity of the challenge to mimic humans, literature often focuses on a single element of a humanoid. For example, Riek and Robinson (2008) focus on facial mimicry of robot using a mechanical chimpanzee head, which is manually controlled to adjust its facial expression based on interaction with humans. Figliolini and Ceccarelli (2002) focus their efforts on mimicking a human index finger and propose a design that could eventually be used for human prosthetics. Riek and Robinson (2008), as well as Figliolini and Ceccarelli (2002) focus on mimicking movement by sending messages to the robot, rather than implementing an AI to learn the movement and then mimic. In much the same way, there exists much literature where the movement of robots is based on how animals move. Birds, fish and insects are used to map out the movement of robots in Kawamura (2010), with Xiuli et al (2006) building a quadrupedal robot based on the movements of a cat. The robots is the aforementioned literature have all be designed to mimic the movement of animals and humans, but do not learn to do so as they hard coded to move in this way.

Atkeson and Schaal (1997) take a different approach to having robots mimic movement in that their research focuses on how robots can learn to move based on demonstration. This is true mimicry whereby a robot observes a human or another robot moving and learns to move in the same way, taking error handling into account (Atkeson and Schaal, 1997). In their research, Atkeson and Schaal (1997) attempt to teach a robot arm how to swing a pendulum horizontally. Multiple techniques are attempted, including directly copying a human who swings the pendulum. In this case, the robot failed to imitate the human as the trajectory used by the robot did not generate enough momentum to swing the pendulum upright. Following this failure, a planning approach was used where the robot was tasked with first finding the correct trajectory using reinforcement learning, and then executing the swing. Atkeson and Schaal (1997) state that the human demonstration of the task provided the robot with a trajectory that identifies the task goal and seeds the learning approach. The robot then learns the task model as well as adjustments that need to be made due to error, which is achieved through reinforcement learning. This enables the robot to carry out the task. Further research related to learning by demonstration has been carried out by Schaal (1997) using the Q learning algorithm to teach a robot how to balance a pole.

While there is no relevant literature pertaining to learning by demonstration in videogames, it is apparent that the AI techniques used in robotics for learning by demonstration are the same as those used in Game AI research, usually to create a better player or more challenging experience.

4.5 Fighting Games

Research pertaining to videogame game AI, specific to the fighting genre, is relatively scarce. However, of the research that has been conducted, the focus is on creating a stronger AI player, or one that adapts its difficulty to cater for the player, providing a level of difficulty corresponding to the human player's own ability. The latter is explored by Ortiz et al (2010). While most commercial fighting games utilise FSMs of some form (Ortiz et al, 2010), the majority of research carried out for this genre pertains to machine learning, however, most commercial fighting games leverage pre-scripted AI techniques such as finite state machines (Lu et al, 2014). A novel problem domain within fighting game AI research is explored by Lu et al (2014), who propose a novel fighting game AI competition platform, where fighting game AIs that have been trained can fight against one another. Such competitions are becoming commonplace at Game AI conferences and include non-fighting games such as Super Mario Bros at the Mario AI Championship (Hisashi, 2014). While the research itself is based on programming a platform to allow custom AI fighters to play against one another, few details on the implementation are offered.

In Cho's (2006) research, the intelligent character is trained using reinforcement learning, which entails assigning a 'reinforcement value' to the action selected by the neural network based on the game scores in an attempt to 'reinforce' the decision. This is in contrast to the ANN used by Chaperot et al (2005), which does not use a reinforcement value. Cho, et al (2006) stated that for the IC (Intelligent Character) to be effective, it must be aware of the OC's (Opponent Character) action patterns. This is to say that the IC must know when the OC is setting up for an attack, by, for example, moving forward twice. Rather than reacting to the action of moving forward, the IC must be aware that this action is part of a pattern, and as such, must perform an output accordingly. To achieve this, past actions are inputted into the neural network. The number of past action inputs is directly proportional to the length of the opponent's identifiable action pattern.

There were various experiments conducted by Cho et al (2006). The trained IC fights a randomly acting opponent, the results of the fight show that the IC scores higher than its randomly acting opponent. However, this cannot be considered a successful application of neural networks as the OC did not behave in an intelligent manner, failing to provide a challenge to the IC.

The second experiment observed the IC's ability to deal with action patterns performed by the OC. Four action patterns to be executed by an OC were selected, and their optimal counter actions were identified. For each of the patterns, 4 IC's were used to fight the respective OC; the first did not use past actions as input and is referred to as simple IC. The second fighter used the last past action as an input and is referred to as Pattern Learning IC1. The third fighter uses the last two actions as inputs (Pattern Learning IC2), and the final fighter uses the last three actions of the OC as inputs (Pattern Learning IC 3). An experiment was conducted to investigate how many times the optimal counter pattern was found. The results showed that generally, the pattern learning IC's far outperformed the simple IC. However, there did not appear to be much of a correlation between the number of past actions used as inputs and the performance of the IC for the pattern learning ICs. It should also be noted that for some patterns, the only IC that could find a counter-pattern of actions was Pattern Learning IC3.

Cho, et al (2006) presented a set of results that suggest a successful implementation of a neural network in a fighting game, however, it could be argued that the research conducted by them predominantly focuses on pre-defined action patterns. The results of the experiments were not placed into the wider context of observing how the neural network performs against a human player. Cho (2006) established that the intelligent character can indeed out-perform a randomly behaving opponent character, however, there is nothing to suggest that it would perform equally well against a human. This is in contrast to the research carried out by Chaperot (2005), which yielded positive results even against a benchmark set by a human player.

The approach taken by Ortiz et al (2010) was to have an AI agent, charged with controlling the opponent fighter, adapt to the human fighter such that learning is carried out offline between rounds. The novelty with this particular research lies in the fact that the goal is not to enhance the AI fighter's ability, but rather adapt it to make for a more interesting experience. The AI agent is split into three subagents, the first of which is referred to as the main subagent, the second being the executing combo subagent, and the third being receiving combo subagent.

Actions are selected from states as the main subagent utilises reinforcement learning. Greater positive reward is issued at the end of each round, as the difference between the health of the human player and that of the opponent player is smaller. Negative reward is issued as the difference between health is greater. Ortiz et al (2010) argued that this is a measure of the level of the human player's ability and how closely matched they are by the AI agent controlled opponent. The executing combo subagent is responsible for selecting appropriate combos based on an algorithm that reads in multiple parameters

and adapts the initial combo set between rounds. The receiving combo subagent is also put to work at the end of each round by mining patterns within the set of combos executed by the user. The Receiving Combo Sub Agent (RCSA) can be invoked during a bout with the human player by the main subagent, allowing the AI controlled opponent to predict combos that are to be executed by the human player, based on the data previously mined. This enables the main subagent to select a combo breaker accordingly (combo breakers are the means to counter a combination of strikes).

After establishing three static players (weak, medium and strong) and two adaptive players based on the aforementioned novel architecture (one of which had 20 rounds worth of previous training), an experiment was carried out to test the effectiveness of the technique. 28 players were engaged and asked to play between 15 and 30 rounds against each of the five AI players (whether the player was static or adaptive was never disclosed to the users). The user was at liberty to stop playing after 15 rounds if they were not having fun. The users completed questionnaires regarding their experience with each opponent and were asked to rank each of the opponents in terms of how much fun they had playing against each AI player.

Ortiz et al (2010) argued that the results showed one of the adaptive players as having the least negative reviews. While this is true, that is not to say that it had the most positive reviews either. The most positive reviews were awarded to the strong static player and, looking closely at the results, it is evident that the two adaptive players had the least negative reviews, with the strong and medium static players very close behind.

The work carried out by Ortiz et al (2010) is novel in its problem domain, and also the architecture of the solution. It is somewhat ambitious as it is using human perception of fun as a success criteria, which ultimately led to poor results. It may be that the game is simply not enjoyable after playing through a large number of rounds against the static opponents, hence why it is conceivable that by the time the users got round to fighting the adaptive players they were already bored of the game in general. Sadly, the order in which the opponents are faced during the experiment is not discussed which would give more credibility to this argument.

Ricciardi and Hill (2008) had different success criteria in that they attempted to create an AI that can adapt online, opposed to the 'between rounds' offline approach seen in Ortiz et al (2010). The primary goals of Ricciardi and Hill's (2008) research was to have the AI adapt to varying strategies carried out by the human player, such that it can learn in real-time during a bout and respond efficiently, making it better equipped to defeat the human. This goal is challenging due to limited training data that can be accumulated and processed online. A further goal was to recognise particular player patterns such

that previous player models could be recalled, offering a better starting position for the AI. While there is acknowledgement that adapting difficulty makes for a more fun experience, a notion that concurs with Ortiz et al (2010), the requirement to vary difficulty online is omitted from the scope of this particular research.

Ricciardi and Hill (2008) utilised an enhanced Markov Decision Process in a self-coded basic fighting game that allowed for four actions besides movement and standing still; punch, kick, throw and block; which are used in a rock, paper, scissors context as described by Gingold (2006). Improvements to the standard Markov Decision Process were made to increase the weights on the most recent transactions, allowing for a more representative dataset where only the most recent transitions were considered. This ensured older data, that may have been repetitive and was inaccurately being represented in the MDP, no longer distorts the data.

Results indicate that the adaptive AI fighter can easily defeat a static state machine based AI fighter. However, by the authors' own admission, this was to be expected as the static AI ran on the same states that formed the basis for the MDP, making it easy to predict. However, results against the human players were also impressive as it won most games. Unfortunately, no statistics were offered to illustrate this. The AI also seemed to learn to counter what were previously considered to be exploits whereby the human player could take advantage of instances where the AI did not know how to respond to its benefit. Following numerous training, the MDP based AI learned how to counter such exploits with basic, then eventually complex, tactics of its own. With regards to the goal of player recognition such that previous experience can be recalled, this proved to work, however, it was deemed unclear whether or not it was feasible to utilise this functionality rather than simply adapting to each player as though it were a new one. The question raised here is predominantly driven by performance and the time it takes to 'swap in' a player.

The work carried out by Ricciardi and Hill (2008) is novel and keeps the focus on improving the skills of an AI in a fighting game, which can be measured by quantitative means. Sadly, no statistics are offered to elaborate on the success of the technique used. Furthermore, it could be argued that the game in question is not representative of a fighting game and makes for a weak test bed. By the authors own admission, the game is rudimentary and only allows for seven actions per player.

Danzi et al (2003) also attempted to develop an AI that utilises reinforcement learning in an effort to allow for adaptable difficulty levels. In this instance, the test bed is a self-coded fighting game called *Knock 'em*. The game presented here is more representative of commercial fighting game than of the one proposed by Ortiz et al (2010) and Ricciardi and Hill (2008). The game itself allows for a wider

range of moves, including weak and strong punches and kicks, as well as the ability to perform jump attacks and crouch attacks. The fighters can also throw projectiles and perform magic which deplete mana, the magic meter, which replenishes as time goes on. As is standard with most fighting games, each fighter has a health bar that is depleted as damage is inflicted.

A tuple of attributes relating to the game is collated. This includes the state of both fighters (crouching, jumping or standing), the distance between the fighters, the amount of mana each fighter has and the projectile status. The state of each fighter, mana available and distance between fighters are attributes that help determine the next move to be carried out. The reinforcement signal is mandated by the health difference between the fighters, which are each initiated at 100, making the range of reward [-100, 100]. Positive reward is given if the AI fighter has more health than the opponent, unlike Ortiz et al (2010), where positive reward was given for minimal difference in health. It is this reinforcement signal that dictates the challenge function where, if the reward is less than -10, the bout is considered easy; greater than +10 suggests the bout is difficult for the human player; whereas zero suggests the fighters are equally matched.

The Q Learning reinforcement learning algorithm Danzi et al (2003) is applied to the game with a fixed learning rate of 50% and discount rate of 90%. Prior to the evaluation, the AI fighter was trained against random fighters in 500 fights. An experiment was carried out whereby a state machine drive AI fighter, a traditional reinforcement learning AI fighter (one that adapts to become better) and the AI fighter described in the paper (herein known as the adaptive RL fighter) each fight three further AI fighters 30 times each. The three further AI fighters consist of a state machine fighter, a randomly behaving fighter and reinforcement learning fighter that plays as best as possible. The average life difference across the 30 bouts are collated and presented. The results show that while the adaptive RL fighter lost most bouts, the difference was minimal, with the average life never dropping below -10 across the three opponent fighters. This is sound evidence that the approach is successful. Other results were expected in the traditional adaptive fighter was easily the strongest fighter by a large measure.

Cho et al (2007) conducted further research into the realms of machine learning and fighting games by comparing techniques within a basic fighting game. The game itself is limited in that there are only five attacks that can be performed, as well as movement, guarding and staying idle. The attacks and movement each take a certain time to execute, where clocks are used to measure this metric. Three different AI fighters were created, each using either neural networks, genetic algorithm or

evolutionary neural network, and fought against an opponent character 10 times, the average of which was used as a metric to measure performance of the AI character.

For the AI fighter that has been implemented using a genetic algorithm, chromosomes include information on the distance between the fighters, the previous actions carried out by the opponent, as well as the current frame of the current action being carried out by the opponent. This information determines the output from the AI fighter. The ANN AI fighter is implemented using the same technique utilised previously by Cho et al (2006). The evolutionary neural network AI fighter was implemented by expressing weights of a back-propagation neural network as chromosomes of a genetic algorithm. It is hypothesised that this would speed up the back-propagation and redistribution of weights.

Experiments were carried out to determine the number of matches against an opponent fighter before the same score ratio was acquired across the three AI fighters. For the neural network, the number of matches can be used as expected, for the genetic algorithm and evolutionary neural network fighters, the number of generations must be converted to the number of matches. As the convergence speed of the evolutionary neural network is the fastest, and its convergence score ratio is second only to neural networks, Cho et al (2007) proclaimed that it is the most appropriate AI technique for fighting games.

Research of particular interest was carried out by Lee (2005) who utilised adaptive learning in a self-coded fighting game, Alpha Fighter, by delegating certain remits to multiple Hidden Markov Models (HMMs). Lee (2005) divided the decision making of AI into various levels; strategic, tactical and operational. Of the four HMMs used, one was used to predict the opponent's attack, two were used at the tactical level, with the fourth being used at the operational level. The strategic layer ultimately dictates whether the player should choose offensive or defensive tactics based on the threat level of the opponent. Tactics are selected by one of the two tactical HMMs, one for offensive tactics, the other for counter tactics. These tactics consist of multiple steps, or operations which are actioned to result in moves carried out by the AI fighter. The results of a survey regarding the adaptive AI suggest that the 10 people that played the game found it challenging and fun to play, more so than a non-adaptive version of the game.

The research conducted by Lee (2005) is another attempt to enhance a fighting game AI in an effort to make it more challenging and more fun, and offers an incredible novel approach to decision making within fighting games by splitting the process on the strategic, tactical and operational layers and passing information between them.

Yamamoto et al (2014) carry out research in the competitive fighting game AI space, where their proof of concept game, FightingICE is used as a platform to test a newly devised AI that aims to defeat other AI players by predicting and then countering their moves. The FightingICE platform is used in competitions to determine superior fighting game AI agents, and largely conforms to the basic principles of a fighting game. There are two on-screen characters which can move in four directions across two dimensions (up, down, left and right), as well as execute three attacks at varying levels (high, medium and low). For the purposes of the research carried out by Yamamoto et al (2014), the same character is used as both on-screen players, such that the moves available are consistent across both fighters. The game deviates slightly from fighting game norms described in Chapter 2 as there is no upper bound to the damage the on-screen fighters can incur. This is to say that the bout goes on for 60 seconds and the fighter who has sustained the least amount of damage during the course of the round is regarded as being the winner.

The AI designed by Yamamoto et al (2014) hinges on an initial, and then ongoing data collection activity. Data including moves the opponent makes, the distance between fighters and the relative co-ordinates of the fighters are collected. By collecting the data, the AI is then able to utilise K nearest neighbour classification to predict the opponent's next move given the co-ordinates of the fighters. Depending on what the value of k is (the experiment trials a variety of values for this variable), the AI simulates possible countermoves against each of the moves the opponent may execute. Of these scenarios, the move which will cause the greatest difference in incurred damage in favour of the AI is selected. The analysis leading to the selection of the AIs move is done using 'brute force', rather than using another technique such as reinforcement learning, which may yield better performance due the reduced computational burden.

Yamamoto et al (2014) use their AI to compete against three different AI opponents that have previously partaken in FightingICE competitions; T, the reigning champion, SejongAI, the runner-up and Kaiju, who took third place in the 2013 tournament. Against each opponent, the value of k was set to 1, 3, 5, 7, 9 and 11. Each AI opponent was played against for 100 matches (each match containing 3 rounds) for each value of k. The average score across the 100 matches for each round and each value of k are presented. The results show that the AI Yamamoto et al (2014) have designed and implemented is able to beat each of its opponents, earning a higher average score by dealing more damage than it incurs for each value of k. It is apparent that some values of k perform better than others across different rounds and opponents. Yamamoto et al (2014) suggest a possible future development would be to adapt the value of k to the opponent and round that is being played out.

Yamamoto et al (2014) have proved that their proposed AI is strong enough to defeat the competition, however, the FightingICE platform seems too basic a test-bed in terms of the moves available. There only appears to be a single parameter; health, which is at play and that isn't even taken into account by AI. The AI is solely analysing position data to predict what the opponent will do. While this is apparently effective, as evidenced by the results, it is highly likely that a human player would consider the health of their fighter and well as the opponent when deciding which move to make, rather than just looking at the positions of the fighters. It would be interesting to see how the AI responds to a human opponent and this may make the training of the AI more complex.

Thunputtarakul and Kotrajaras (2007) carried out research closely linked to the aims and objectives of this Thesis in their attempt to establish a ghost AI that plays tactically as a human would play. Furthermore, they attempted to achieve this in a commercial game, *Street Fighter Zero 3* (Capcom, 1998). It should be noted that the work carried out used a hacked, emulated version of the game in conjunction with an AI engine AI-TEM. The ghost AI is created by creating a data file containing information on frames of animation enacted by the player to be mimicked (caused by performing actions) and recording the conditions under which these animations took place. Short lived animations are removed from this database as they are perceived as being unimportant and anomalies within the data. This information is consolidated by pairing up actions carried out to the conditions in the game under which they were carried out, and then encrypted into 32 bit strings. Each action has a 32 bit string, the first 8 bits of which contain information on the action being carried out by the player. The remaining bits contain information regarding the game state / circumstances under which the move was executed. These include the following:

- The distance between characters in two dimensions;
- The state of the opponent (whether or not the opponent is attacking);
- Whether a player induced projectile is on the screen;
- The distance of projectiles on the screen;
- Damage inflicted to the opponent in a single animation frame at a specific point in time;
- Whether the player is facing left or right;
- Whether the player is in a corner;
- Whether the opponent is in a corner.

At its core, the technique used by Thunputtarakul and Kotrajaras (2007) relies on scanning through the data file (referred to as a battle log) and finding matches in the current game based on the criteria

above (which is articulated via the last 24 bits of the 32 bit string so as to minimise processing overhead), and then executing the frames of animation recorded in the first 8 bits of the string.

To evaluate the technique, an experiment was carried out whereby 32 participants were asked to play the game between 2 and 10 minutes in training mode, where the health of the on-screen fighters did not deplete, nor did the super combo meter which typically allows the player to unleash powerful attacks. Data from this bout, against what is presumably an AI controlled fighter (the publication does not discuss the nature of the opponent), is then fed into the battle log and the player is asked to 'semi-play' a further two times. This means the player watches the bout between their ghost and opponent, while simultaneously playing the game against the opponent again.

Two separate methods are used for splitting the 'controller signals', which are the action signals telling the AI to perform an action and animate the on-screen characters. The experiment captured these controller signals and compared the player and AI signals, as well as the players' signals with their own data from a subsequent play through. Of the two methods for splitting the controller signals, the most successful relied upon capturing data every 15 frames. Across the 32 players, when compared against the play styles of the human players, the ghost AI only yielded a 26% average match. However, that being said, the players' own second play through only maintained an average of a 35% match. These results are indicative of the ghost AI not performing well, and/or the experiment not yielding quality data to feed into the AI.

The experiment itself was flawed in that the participants were forced to 'semi-play' where they must divide their attention between playing and observing. The observation could influence how they play the game and, judging by the results, distracts them from playing out the same style. The fact that the ghost AI only accomplished a 26% match suggests that its performance is poor, but the more alarming result is the fact that the human players never really played using the same style more than once, with the average match being 35%. The participants gave an average satisfactory score of 72%, suggesting they were largely satisfied their ghosts were doing their play styles justice. However, this is not entirely credible as these participants also believed they were playing their own style through accurately multiple times, when in reality it was deemed that the average match for this was 35%.

When we examine the data encrypted in the bit string, this is not all that surprising. The battle log ultimately contains contextual data that are detailing very specific events. The level of granularity would make it difficult to replicate circumstances and verify with certainty that should those circumstances come into fruition again, the player would act in the same way. For all the emphasis placed on positioning, opponent states and so on, it could be argued that the most influential factors

in determining how the player behaves - the players' and opponent's health -, is not even considered. Both the health meter and super combo meter are not factors in the bit string, making their values meaningless which, in the context of this research, they are by default, as the experiment takes place, strictly within the game's training mode.

Unfortunately the health and super combo meters are not considered, as they would have offered a strategic approach to game play. Instead of considering these strategic elements, the technique focuses only on reacting to a given situation. This makes the underlying assumption that players' actions are entirely driven by the environment they are in (besides from arguably the two most important features of the environment, the health and super combo parameters). The danger in making this assumption is that it does not consider a player ignorant to their opponent's actions, who prefers to use their own variety of attacks when they feel it is suitable. Considering the AI seems to be geared towards playing strictly on an operational level, this is quite an oversight.

The work carried out by Thunputtarakul and Kotrajaras (2007) is commendable in that it offers a means of accessing commercial fighting games AI. The approach of consolidating real-time factors during bouts played by a human, and then replicating their actions when the circumstances occur is novel and could be built on using more pertinent game parameters that influence decision making. This research is taken further by Lueangrueangroj and Kotrajaras (2009) who build on the use of AI-TEM using an emulated version of Street Fighter Zero 3 Upper. Lueangrueangroj and Kotrajaras (2009) build the ghost AI in the same way Thunputtarakul and Kotrajaras (2007) have, but in this research the focus is not only on imitation but also efficiency in terms of leveraging the possible actions to deal the most damage to the opponent.

The work carried out by Lueangrueangroj and Kotrajaras (2009) splits the generation of the AI into two distinct processes; the imitation process and learning process. The imitation process focuses on developing the ghost AI in much the same way as Thunputtarakul and Kotrajaras (2007) by using parameters including the distance between fighters, distance between projectiles and the AI fighter, and the AIs current action. Should the game state yield a situation where these variables are the same as ones in a previous bout featuring the human that is being imitated, then the AI would carry out the same move as the human did in the previous situation. Based on Lueangrueangroj and Kotrajaras (2009), it appears that this is calculated as a simple look-up rather than a classification problem. The addition in this research is the learning process, which adds a weighting to each move that is carried out by the AI, to quantify how effective it is at dealing damage when it was executed. When the AI

fights an opponent, it imitates the human player that it is based on, but also selects moves based on the weighting, which adjust online as the AI learns making for an adaptive AI.

The work carried out by Lueangrueangroj and Kotrajaras (2009) certainly takes the previous work of Thunputtarakul and Kotrajaras (2007) further by developing an AI that rather than blindly imitating a human based on static data in a file, adapts online and leverages the moves available in a manner that allow it to become a more efficient fighter. However, it could be argued that this is something of a paradox as the AI cannot imitate the human, and play better than the human would at the same time, as it would have evolved into a fighter beyond the human fighter's capabilities. Also, much like Thunputtarakul and Kotrajaras (2007), there seems to be no consideration for the game environment beyond the distance between the two fighters. The imitation process does not consider what the opponent is doing, nor the previous moves that have been carried out and the in-game statistics of both fighters. This is unfortunate as the game being used as the test bed, Street Fighter Zero 3 Upper, utilised multiple parameters per on-screen fighter, encouraging the player to fight strategically.

4.5 Chapter Summary

In this chapter, a concise survey of literature pertaining to AI in videogames, and particularly, fighting games, is presented. Techniques have been examined in terms of design, as well as usage within the field of Game AI. Having reviewed this literature, it seems that the majority of research done within the field of Game AI utilised Machine Learning, oppose to more static techniques such Finite State Machines. With regards to usage, research within the field of fighting games is very limited, particularly that pertaining to strategies and tactics within fighting games. Research related to mimicking player styles is limited across fighting games as well as other game genres. Based on the literature review conducted, there appears to be a distinct lack of research conducted in the field of AI applied to strategic fighting games. While the use of AI techniques make for engaging Real Time Strategy games as demonstrated by Miles et al (2007) and Barton (2014), the work carried out by Cho et al (2006), seems to be limited to shorter term tactics. Cho et al (2006) implemented ANNs in a fighting game, but the responses of the AI fighter was limited to short term tactics. Further to this, Cho failed to test the AI fighter against a human opponent, making it difficult to gauge the success of the aforementioned technique. Implementing an AI technique in a fighting game to enable the AI controlled player to learn and mimic human strategies is an area of videogame AI that has not yet been explored.

Other ventures into implementing AI in fighting games have focused on either adaptive difficulty settings for increased pleasure during play (Ortiz et al, 2010; Danzi et al, 2003), or on simply enhancing

the AI fighter's ability to make it a better player (Ricciardi and Hill, 2008). Each of these implementations was accessed using independently coded fighting games (non-commercial), which, with the exception of Danzi et al (2003), do not adequately replicate the choices available in modern commercial fighting games. Graepel et al (2004) addressed the issue of the lack of a representative test bed head on, as they implemented reinforcement learning in a commercial fighting game, Tao Feng on Microsoft Xbox. Unfortunately, none of these implementations address the problem of mimicking a human player's strategy. Thunputtarakul and Kotrajaras (2007) attempted to solve the mimicking problem, but they yielded poor results and were restricted to the operational level, taking no account for the strategic elements of the videogame in question, *Street Fighter Zero 3* (Capcom, 1998).

The wider field of AI in videogames provides examples of human players playing against AI controlled players, such as those seen in chess. However, much of the research conducted in the field of videogame AI restricts the evaluation of techniques to results from hand coded (and sometimes randomly behaving) opponents. As videogames AI research stands currently, there is a gap in the field with regards to the use of AI techniques in strategic fighting games, in particular, evaluating the effectiveness of techniques against human players. Key literature that has been reviewed in this chapter pertaining to the application of game AI techniques has been summarised in Table 4.1 below.

Table 4.1 – Literature Review Summary

Technique	Source	Summary	Conclusion
Supervised Learning	Chaperot et al (2005) Chaperot et al (2006)	The aim of the research was to implement an ANNs in the Motocross racing game such that the bike is ridden as well as possible, but while still retaining the characteristics of a human playing the game. An experiment where a track was completed by a human, ANN AI and genetic algorithm AI showed that the ANN performed nearly as well as a human player and outperformed the genetic algorithm.	Literature related to the usage of supervised learning in videogames has generally yielded positive results. There are many techniques that have been utilised to enhance AI players, creating more of a challenge to humans, and also in terms of identifying human play styles. However, there has been no research conducted in terms of
	Cho et al (2006)	Using ANNs to create a superior AI controlled player in a fighting	

Technique	Source	Summary	Conclusion
		game. Back-propagation is used in conjunction with feed forward networks to create intelligent fighters that outperform AI controlled fighters of a static nature. There is no benchmark against a human player however.	mimicking human players.
	He et al (2008a)	He et al (2008a) used Pac-man as a test bed to investigate the use of Bayesian Belief Networks and Naïve Bayes classifiers for player modelling. NBC and BBN are found to both be useful for predicting player strategies.	
	He et al (2008b)	He et al (2008b) use Radial Basis Function (RBF) classifier in Pac Man to predict player strategy patterns. He el al (2008b) have demonstrated that the RBF is a powerful classifier for pattern recognition, and have demonstrated in this instance that it is more powerful than BBNs and NBC.	
	He et al (2008c)	He et al (2008c) used the videogame of Dead End as a test bed for their approach to player strategy pattern recognition. KNN and BBN are used to successfully classify player strategies. KNNs perform well, but BBNs yield better results in an experiment conducted using Dead End.	

Technique	Source	Summary	Conclusion
	Cho et al (2007)	Cho et al (2007) conducted further research into the realms of machine learning and fighting games by comparing techniques within a basic fighting game. As the convergence speed of the evolutionary neural network is the fastest, and its convergence score ratio is second only to neural networks, Cho et al (2007) proclaimed that it is the most appropriate AI technique for fighting games.	
Clustering	Anagnostou et al (2009)	Applying the CURE algorithm to a space invaders game to determine play styles. The CURE algorithm successfully classified two of the game features synonymous to the two player types.	Clustering is a powerful technique that is capable of organising data into as yet undefined sets. Its application to videogames has provided positive results.
Player Modelling	Drachen et al (2009)	Drachen et al (2009) have explored player modelling in the commercial game Tomb Raider Underworld (Crystal Dynamics, 2008), using self-organizing maps, which is a form of clustering. The approach is successfully used to model how players play the game.	Player modelling is a technique that has been demonstrated to have uses within commercial games, particularly with regards to identifying player preferences and strategies. This approach has relevance to the research conducted in this Thesis as strategies must be identified
	Pedersen et al (2009)	Pedersen et al (2009) exhibit player modelling in a Super Mario Bros clone called Infinite Mario Brothers. Pedersen et al concluded that by using non-linear preference modelling, given a set	

Technique	Source	Summary	Conclusion
		of game metrics (inputs), ANNs can be trained to predict the emotional response of the player.	before they can be mimicked.
Reinforcement Learning	Ortiz et al (2010)	Reinforcement learning is used to adapt an AI controlled player in a fighting game to make the game more interesting for the human individual that is playing. Ortiz et al (2010) argued that the results showed one of the adaptive players as having the least negative reviews. While this is true, that is not to say that it had the most positive reviews either.	Reinforcement learning has been used to create stronger and more interesting AI fighters within basic fighting games. While the genre is of relevance to the research carried out in this Thesis, the objective of creating an interesting or enhanced AI fighter does not fall in alignment with the aims and objectives of this Thesis.
	Danzi et al (2003)	Danzi et al (2003) attempted to develop an AI that utilises reinforcement learning in an effort to allow for adaptable difficulty levels. The results of an experiment show that while the adaptive RL fighter lost most bouts, the difference was minimal, suggesting that the approach is successful.	
Markov Models	Ricciardi and Hill (2008)	The primary goals of Ricciardi and Hill's (2008) research was to have the AI adapt to varying strategies carried out by the human player, such that it can learn in real-time during a bout and respond efficiently, making it better equipped to defeat the human. Ricciardi and Hill (2008) utilised an enhanced Markov Decision Process in a self-coded basic fighting game that allowed	The work of Ricciardi and Hill (2008) and Lee (2005) have provided great insight in how Markov models can be used to enhance fighting game AI. Once again, the focus of the research conducted here pertains to using these techniques to enhance the player

Technique	Source	Summary	Conclusion
		for four actions besides movement and standing still. Results between the adaptive AI and the human players were impressive as the AI won most games. Unfortunately, no statistics were offered to illustrate this.	experience by providing more of a challenge, oppose to identifying then mimicking a human strategy.
	Lee (2005)	Lee (2005) attempts to enhance a fighting game AI in an effort to make it more challenging and more fun, and offers a novel approach to decision making within fighting games by splitting the process on the strategic, tactical and operational layers and passing information between them. The multi-tiered HMM architecture was perceived by the player to enhance the intelligence of the AI fighter.	
Hard Coding	Thunputtarakul and Kotrajaras (2007)	Thunputtarakul and Kotrajaras (2007) carried out research to create a ghost AI capable of mimicking human playing styles using a hacked, emulated version of Street Fighter Zero 3 in conjunction with an AI engine. This is achieved by replicating the frames of action played out by the characters. The approach is based on coding routines that scan through the a file (referred to as a battle log) and finds matches in the current game based on a specified criteria, and then executing the frames of animation	The hard coding approach of Thunputtarakul and Kotrajaras (2007) is noteworthy in that the objective is closely related to that of this Thesis in that the human 'play styles' are to be mimicked. However, the means of achieving this objective are by mimicking animations irrespective of the in-game statistics. This, coupled

Technique	Source	Summary	Conclusion
		recorded in the first 8 bits of the string. Given that during the experiment, the health and super combo meters did not deplete, and the participants were asked to 'semi-play', further work would need to be carried out to produce more meaningful results.	with the 'semi play' evaluation approach do not provide a firm foundation on which the research of this Thesis can be based.

In conclusion, there is a distinct lack of research related to the application of AI in fighting games and the problem domain of mimicking human strategies in fighting games using AI is completely uncharted in terms of academic research. However, the research reviewed in this chapter has provided novel ways in which a variety of AI techniques can be utilised to solve this problem. Having come to this conclusion, the problem domain can be detailed further and an appropriate technique can be designed to solve the problem.

Following the literature surveyed in this chapter, a significant gap has been identified in the field of game AI, pertaining to mimicking human strategies and tactics in fighting games. The approach to fill this gap through the research conducted in this thesis is described.

The literature review yielded results indicative of a lack of research related to the field of Game AI. In particular, the sparse research in this field gave rise to a significant gap related to the usage of Game AI to solve mimicking problems, especially in fighting games. Having established this gap and described the approach to filling this void, the research is able to progress by building multiple solutions to address the problem of mimicking human strategies in fighting games.

The research is now able to progress by designing and implementing the necessary tools and peripherals to support the solution. This includes the design and development of the solution itself, as well as the proof of concept game that forms the test bed for the evaluation of the solutions. These components are documented in Part II of this Thesis.

Part II – Analysis, Design and Development

Chapter 5 – Proof of Concept Game

5.1 Introduction

To support the development of a solution to the problem, a proof of concept kickboxing fighting game was coded and used as a test bed to evaluate enhancements made at each level. The research utilised a basic prototype proof of concept game representative of commercial fighting games, and focussed on mimicking decisions made at the strategic level. To test this in a proof of concept game, an entire end-to-end solution was implemented. The proof of concept game allowed for greater complexity in strategies that can be used by the human players.

This chapter provides the detailed design for the game, with a justification for the limitations in functionality in that the game needs to be designed to cater for gamers, while allowing the solutions proposed in this theses to be tested. The game's rules and character moves are described. The implementation details of the game are also provided, including tools and digital assets (these include the 3D character models and 3D environment) used to develop the proof of concept game.

5.2 Design

The design of the proof-of-concept game needs to account for the rules of modern kickboxing fighting games as discussed in Chapter 2. In an effort to develop a proof-of-concept game representative of modern fighting games, key gameplay mechanics must be factored into the design to ensure the game meets gamers' expectations by conforming to traditional fighting games (as discussed in Chapter 2), as well as providing specific rules to allow for the testing of solutions provided in this thesis. This section describes and justifies the rules and design decisions.

5.2.1 Game Rules

The proof-of-concept game takes its design cues from traditional fighting games as described in Chapter 2. The game features two on-screen characters, the fighters, who carry out kickboxing moves to inflict damage upon one another. Moves include a variety of punches, kicks, blocks, lunges and evasions. As the characters inflict damage upon one another, their health meter deplete until it reaches zero, at which point the winner is declared. The bout takes place in a boxing ring, making for an aesthetically appropriate background. The game is deliberately grounded in reality and is based on the sport of kickboxing, rather than other fighting games discussed in Chapter 2 that rely on fantasy based projectile attacks and gravity defying moves.

Some features that are not commonly found in such fighting games have been omitted in favour of simplicity. The game involves two fighters who are on screen at all times. As the purpose of this research is to design and evaluate a means of mimicking human strategies within a fighting game, it is important that the game lends itself to strategic gameplay and offers players enough flexibility to devise their own strategies. This is achieved by offering a wide range of moves and implementing features such as multiple parameters per character.

To allow for a variety of strategies, the multiple parameter model is utilised, similar to that found in *Street Fighter Alpha 3* (Capcom, 1998) (see Chapter 2 for further details and screenshots). The main parameter, as in all fighting games, is health. Therefore, each fighter has a health meter, initiated at 100 health points (HP). As a fighter inflicts damage onto their opponent, the health of the fighter receiving the damage is depleted. Once the health of a fighter reaches zero (0 HP) then the bout is over and the opponent is declared the victor. As is the case with all fighting games, different moves incur varying amounts of damage, and the range from which they are effective also varies.

Each fighter has two more parameters to add a level of realism to the game, and further encourage long term strategic play. First, a fighter is equipped with a stamina meter which is initiated at 100 points. As a fighter performs moves or blocks their opponent's attacks, their stamina meter is depleted. Once the stamina meter reaches 0, the fighter cannot perform any further moves, nor can they block their opponent's attacks, ultimately making their defeat inevitable. The inclusion of this parameter forces players to be economical with their moves and not simply go on the outright offensive as this may lead to attacks not connecting, fruitlessly depleting their fighter's precious stamina. It also prevents players from being overly defensive, as this too would be a losing strategy. If a player is blocking excessively, their stamina shall deplete rapidly, eventually making them vulnerable and defenceless against their opponent's attacks.

The third parameter is Morale. The game allows each fighter to dodge their opponent's attacks. There are various dodges for the variety of moves and the correct dodge must be used for the incoming attack if damage is to be avoided. Provided the timing is correct, and the appropriate dodge move is performed, no damage is incurred by the defender, while the attacker loses stamina. The Morale meter is initiated to 50 points and increases incrementally as the fighter successfully dodges their opponent's attacks. As the morale goes beyond a threshold of 75 points, the attacks carried out by the fighter deals double the amount of damage as they normally would. The rationale behind this design decision is to provide players with a risk vs. reward approach to winning bouts. While the prospect of dealing double damage on an opponent is a seemingly a quick way to victory, the timing

to boost the fighter's morale must be impeccable. Should a fighter perform the wrong dodge, or miss-time their dodge by a fraction of a second, they run the risk of needlessly incurring damage.

The game has no time limit, the only way bouts can finish is a fighter either completely losing their stamina or completely losing their health. This allows players more time to strategize and plan their attack, as well as respond tactically to their opponent's actions. This encourages players to engage in a way that lends itself to testing the solutions proposed in this thesis. Movement has been restricted to a two-dimensional plane for simplicity. Fighters can travel towards and away from each other, with the maximum distance between fighters being restricted to the size of the ring, which is 13 units. The traditional jump and crouch functions found in most fighting games are supplemented with lunges (both towards and away from the opponent) and the ability to perform low attacks from a standing position.

5.2.2 Character Moves

The game offers a variety of standard moves, as well as a special move, the haymaker, which deals an exceptional amount of damage, but has a high cost in terms of execution speed and stamina. Table 6.1 below provides a list of moves that can be executed by the player, along with the impact against their opponent.

If the opponent is within the range specified by the 'to' and 'from' attributes listed, and is not blocking or performing an appropriate evasion, they will be struck and their health value decreases. The unit of measurement for distance is based on the in-game metrics are all relative to each other. If the opponent performs a block (or in some cases a low block) when the move connects, their health shall deplete as indicated by the value 'blocked' field. If timed correctly, certain moves can be evaded. For example, if a fighter throws a jab and the opponent performs a 'back' move with the correct timing then the move will not connect and no health will be depleted.

The design allows players to combine their own unique tactics to form longer term strategies. The variety of moves includes lunging forward and back, making for flexibility in movement. This footwork, combined with evasion maneuvers and attacks, make for a creative fighting system, empowering the players to define various strategies and providing them with the tools to execute short term tactics to accommodate said strategies. Further to attacking, players can perform low or high blocks and a variety of evasions. Certain attacks are blocked by using the high block, while others can only be blocked using the low block. The various moves and their effects within the game are presented in Table 5.1.

Table 5.1 – Proof of Concept game moves

Move	From	To	Health	Stamina	Morale	Blocked	Evasion	Notes
Jab	4.1	5	1	1		Stam - 1	Back	
Cross	4.1	5.5	2	2		Stam – 1	Left	
Right hook	4	4.7	3	2		Stam – 1	Back	
Left hook	4	4.7	3	2		Stam – 1	Back	
Uppercut	0	4	4	2		Stam – 1	Right	
Haymaker	4	4.5	10	5				Unblockable
Right body	0	4	2	1		Stam – 1		
Left body	0	4	2	1		Stam – 1		
Short jab	0	4	2	1		Stam – 1	Back	
Short cross	0	4	3	2		Stam – 2	Left	
Evade back					2			Evasion
Evade left					2			Evasion
Evade right					2			Evasion
Push	0	4	2	1		Stam – 1		Pushes opponent 5 back
Block								Blocks high
Low block								Blocks low
Low kick	0	4	2	1		Stam – 1		
Sidekick	4.1	5.5	4	2		Stam – 2		
F Lunge				5				6 Forward
B Lunge				5				6 Back

5.3 Implementation

This section describes how the proof-of-concept game has been implemented and discusses tools, assets and bespoke coding that was compiled to support the development of the game. An overview of the control scheme is also provided.

5.3.1 Digital Assets

The digital assets used in the proof of concept videogame were either developed specifically for the game, or were re-used from freely available example code. The environment in which the game is played comprises a bounded two-dimensional play area. To represent this in an aesthetically pleasing manner, a suitable three-dimensional model would need to be placed on the screen and remain static to give players a clear indication of the bounds along the X-axis. The 3D model of choice was a boxing ring model that was made available via source code from Panda3D (Carnegie Melon, 2010).

Other digital assets include the character models for the fighters that are displayed on-screen. While the game is routed to a two-dimensional plane, the use of three-dimensional character models is appealing due to the flexibility of animation and is less dependent on time consuming frame-by-frame drawings that are typically associated with two-dimensional sprites. For these reasons, the on-screen character models for the fighters were rigged and animated in 3D. A single asset was developed and then re-used for both fighters, with a slight change in colour being used to distinguish the two fighters (often referred to as a pallet swap). The animations and moves across both fighters are the same.

5.3.2 Tools and Coding Overview

The decision to develop the proof of concept game from the ground up was predominantly based on the fact that for the research to be meaningful, a game representative of traditional commercial fighting games would need to be utilised as a test bed. As source code for commercial fighting games could not be accessed, nor edited to implement custom AI, the decision to develop a custom game was made.

To aid in the development of the proof of concept game, several tools and coding libraries were used. The underlying game engine was Panda3D (Carnegie Melon, 2010) which is a 3D game engine that supports Python and C++ code. For the purposes of this research, Panda3D was used in conjunction with Python 2.6. Character models were created and animated in Blender (Blender Foundation, 2013), and exported to Panda3D using Chicken Exporter (2006). Textures to these models were applied using the built-in capability of the Panda3D game engine. Panda3D and Blender were selected because of their ease of use and the speed at which results could be obtained. The focus for the proof of concept

game was to provide a test bed for the research, rather than creating an attractive animation fighting game experience. In this context, Panda3D and Blender were the ideal candidates.

Two players can play the game using PlayStation 3 Sixaxis control pads that had been configured to run on a PC running Windows 7 or Windows 8. This is achieved by using MotioninJoy drivers (Motioninjoy.com, 2012) to recognise and calibrate the hardware. The calibrated control pads can then be integrated into the Panda3D code using the pyGame library (PyGame.org, 2008) and pyPad360 (Rabid Squirrel Games, 2010) python class. The use of the PlayStation 3 Sixaxis controller means the aforementioned tools need to be used. The Sixaxis controller is used because gamers have become accustomed to its feel. This is an important factor because gamers were invited to take part in an experiment to evaluate the performance of the AI solutions. The use of the PlayStation 3 Sixaxis controller is also driven by the fact that there are a large number of buttons, allowing for a wide range of moves to be mapped to commands.

5.3.3 Game Display

The game display was designed to show both fighters at all times, as well as pertinent information such as the values of each parameter. With each character having three parameters, a total of six parameters would need to be displayed, as well as a seventh; the distance between both fighters. This seventh parameter would assist players in making decisions with regards to which move to execute, although this could also be a judgment call. Both fighters are rendered sideways-on and information is shown at the top of the screen so as not to obscure the view of the action. The fighters are displayed using the same 3D model, but with different textures and shades, making them easy to distinguish. Figure 5.1 shows a screenshot from the proof of concept game.

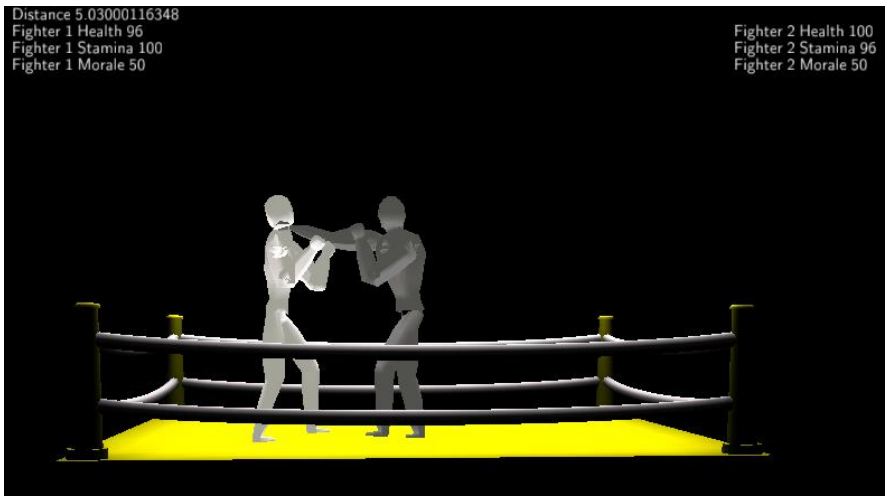

Figure 5.1 – Screenshot of Proof of Concept Game

5.4 Usage of PoC Game

The Proof of Concept game is leveraged in this research by having two human players play the game against one another, where the first player adopts a pre-defined strategy, of which there are ten in total, and plays against the other human player three times. Videos for each of the three bouts are recorded, and transcripts of these bouts are captured and archived for reference. The data for each strategy is then processed by each of the two solutions, giving rise to two AI agents per strategy. The second of the two human players of the initial three bouts then plays against each of the AI controlled opponents. These two bouts are also recorded, making for five videos per strategy; three videos featuring human vs. human bouts, one video featuring the human vs. k nearest neighbour AI fighter and one final video featuring the human vs. data driven finite state machine (DDD FSM) AI fighter. Transcripts of the human vs. AI bouts are also captured and archived.

Having collected a total of five videos for each of the ten strategies documented in Chapter 8, the prerequisites for the evaluation are in place. A group of ten observers, each of whom are well versed in fighting game mechanics, are asked to view one of the human vs. human videos for a given strategy, and are also provided with a high level description of the strategy. This is done to provide each observer with a frame of reference and familiarise them with how strategies are played out in the proof of concept game. Each of the observers is then shown a further four videos for the same strategy comprising of the following:

- Two videos of the human vs. human bouts
- One video of the human vs. KNN AI bout
- One video of human vs. DD FSM AI bout

In the case of the four videos stated above, the observers are not told which of the two on-screen fighters is executing the strategy, nor are they told whether the strategy is being executed by an AI or human player. Each observer is then asked to record which of the following three scenarios they feel holds true for each video:

- Strategy is being played out correctly by a human player.
- Strategy is being played out correctly by an AI player.
- Strategy is not being played out correctly.

Results are collated for each observer against each of the four videos for each strategy, giving rise to 400 interpretations (10 observers, 10 strategies, 4 videos per strategy). Of the 400 observations, 200 are for videos featuring human vs. human bouts of the 10 strategies. This set of 200 observations

serves as the control group as it is telling of whether or not the group of observers are capable of recognising the strategies in question being played out by humans. The number of observers was set to 10 in an effort to ensure only people well versed in fighting games and how to play strategically partook in the observations so as not to skew the results. Broadening the group of observers was considered when designing the experiment, however, due to constraints in terms of understanding how fighting games should be played out, and what it means to have a pre-determined strategy, it was decided against to avoid to the inherent risk of nullifying the results of the experiment. The 10 observers, and their 400 observations make for a sizable set of data given the quality of the observations, which can be attributed to the familiarity the observers have with fighting games and martial arts strategy. Any shortfall in terms of the number of observations was mitigated by the quantitative analysis of the transcripts produced in the human vs. AI bouts.

This approach to evaluation has been adopted to firstly ensure adequate data are being captured during the human vs. human bouts and fed into the AI solutions. The experiments also yielded qualitative results from the observations, which could be backed up by quantitative results from the transcripts if required. The results of the experiment indicate which solution was perceived to have successfully mimicked the strategy, and whether or not this was done in a manner so convincing that it could be passed off as being executed by a human player. This approach also ensures that the ability to mimic a known strategy is being assessed, rather than how good the AI player is. A Turing Test was considered as a means of evaluating the AI solutions. However, this would be more telling of whether or not the opponent was an AI or human, rather than whether or not the opponent was playing out a human strategy in a manner that the human would have played it.

Ten strategies were selected as this number can adequately encompass a representative sample of strategies that can be executed within the constraints of the proof of concept game. The ten strategies, that have been selected and documented in Chapter 8, exhaust the available move-sets of the proof of concept game, and cover a wide range of play styles and approaches. Additional strategies could have been factored into the experiment, but these would have been derivatives of the ten established strategies and would not necessarily add further value. The decision to have ten observers was based on the fact that a large number of observers, coupled with a variety of strategies, would make it easier to determine whether or not there were trends in the observations, and potentially pinpoint problematic strategies that are difficult to mimic.

In terms of success criteria, the primary focus was on determining whether or not the strategies being played out by the AI were perceived as being played out accurately. Against this criteria, for the

solutions to be considered fit for purpose, the expectation is that at least 75% of strategies shall be perceived to have been mimicked successfully (per solution). The perception of whether or not they are played out by a human player or an AI is not necessarily a critical success factor for this research, but this metric is telling of how convincingly a strategy has been mimicked. This 75% success criterion is based on the nature of the strategies outlined in Chapter 8. It is expected that for each solution, there are two strategies which will prove challenging to mimic due to the nature of the moves being executed, which accounts for 20% of the overall result. A further 5% contingency has been added in light of potential errors that may occur during the observations, however, this number is relatively low due to the skill level of the observers. This brings the hypothesised error to 25%, yielding an expected 75% success rate.

5.5 Chapter Summary

In this chapter, the proof of concept game has been described in terms of the core game mechanics and rules. The tools used to build the game have been identified and the limitations and abilities of the fighters within the game have been discussed. A rationale has been provided for key design and implementation decisions. The resultant design conforms to the rules present in traditional fighting games, hence meeting gamers' expectations. Further to this, the multi-parameter approach puts additional emphasis on planning and executing strategies during gameplay.

The proof of concept game developed for this research conforms to the traditional fighting game design described in Chapter 2. The proof of concept game is a two-player fighting game, with each fighter having three parameters; health morale and stamina. Each fighter can execute a wide range of moves that have varying effects on themselves and their opponent. The game has been designed and implemented such that it conforms to traditional fighting games, while offering players enough flexibility to create their own strategies and tactics.

The evaluation high level design has been provided as part of this chapter. The approach to acquire data based on the perceptions of numerous observers is being used as it is a means of understanding whether or not each solution is fit for purpose in terms of mimicking a known strategy, rather than evaluating whether the game is being played by a human or AI player. Additional details on the experimental design, including the strategies that have been selected to be mimicked, can be found in Chapter 8.

Chapter 6 – K Nearest Neighbour Solution Architecture and Design

6.1 Introduction

The research conducted in this Thesis contributes to the field of videogame AI by filling the gap identified in Chapter 4. In order to achieve this, further research on existing techniques and their applications has been carried out as part of this thesis. The techniques that were researched were then implemented in a strategic fighting game, where the AI player learns to mimic the human player actions. The key factor in this research is the evaluation of techniques against human players. The overall deliverable is a robust technique that can be applied to long-term strategic fighting games, enabling the AI controlled player to perform using the same strategies as the human player it learned from, based on the in-game conditions.

Having conducted a literature review, and gained an appreciation for game AI techniques, as well as those evidenced to have been implemented in fighting games, two separate solutions have been designed and developed. The first solution utilises k nearest neighbour classification and is detailed in this chapter.

The research conducted here to solve the problem, and answer the overarching research question 'How can existing Game AI techniques be used to successfully mimic basic human player strategies in strategic fighting games?', began by implementing a prototype based on what has been learned from the literature survey. The purpose of examining key techniques during the literature review in Chapter 4 was to acquire knowledge sufficient enough to create a prototype solution to pave the way in answering the aforementioned research question.

Having established the problem domain, as well as the environment within which a solution can be tested, the next step is to define the solution itself. This chapter provides a detailed description of the architecture and design of the first of two solutions to the problem.

6.2 System Architecture

This section provides a high level design of the architecture for the k nearest neighbour solution, as well as the rationale for the key design decisions. This section describes how the architecture hangs together and how it was used to mimic human player strategies. This approach is solely reliant only on selecting the most appropriate operation from a single pool of moves, building on concepts established by Thunputtarakul and Kotrajaras (2007) in that decision making is done strictly at the operational level. However, unlike Thunputtarakul and Kotrajaras (2007), the design of the k nearest

neighbour solution is concerned with pertinent in-game parameters which were driving factors for the behaviour and strategies exhibited during the human vs. human bout.

6.2.1 Context

Before establishing the design of the solution, the context of its usage must be defined. As with many AI solutions discussed in Chapter 4, the results that can be produced are heavily dependent on the data being fed into the solution. For example, a Naïve Bayes Classifier must first be trained before it can produce quality results.

The solution presented in this chapter is designed with strategic approach to playing fighting games in mind. The solution is reliant on the proof of concept game from Chapter 5 which encourages gamers to play the game using the variety of moves available and formulate their strategies ahead of playing. The k nearest neighbour solution is reliant on two human players playing against each other a number of times, using the same strategy each time. This means that players should perform the same actions each time under the same situations as mandated by the values of each parameter. These data are recorded in real-time and are transcribed and saved to a file following each bout. This is a prerequisite to processing data and producing meaningful results via the k nearest neighbour solution, and is referred to within this research as the *data capture phase*. Once the data capture phase has been completed, the data are fed into the solution, which is then put into action during a bout between a human and the AI player, which is controlled by the k nearest neighbour solution AI. This is known as the *execution phase*.

The code for the k nearest neighbour solution AI has been developed to mimic the first fighter (on the left hand side of the screen). During the execution phase, the first player shall be mimicked by the AI fighter, who shall appear on the right hand side of the screen (traditionally where the second player would be).

6.2.2 Operational Data Capture and Analysis

The first step in mimicking a human player is collating the necessary data from the human vs. human bouts. This is achieved by recording the data in real-time and then spooling these data to a file. Pertinent data to be spooled includes the action made by each player, and the parameters at the time of performing the action. While data against moves performed by both players are recorded, the focus is on the first player (left-hand-side of the screen). Table 6.1 below shows a sample of data recorded during a bout between two human players.

As shown in Table 6.1, data are spooled whenever an action is performed by either player, with the move column containing a single letter identifying the action made. The second player's moves are

also recorded and are prefixed and suffixed with an asterisk (*). Each entry has a timestamp and play-through number, which signifies the iteration within the same strategy is being played. These two attributes combined form a primary key for these data. For the research conducted in this Thesis, each strategy is played three times. Other data collected include the six in-game parameters, player 1 health, player 1 stamina, player 1 morale, player 2 health, player 2 stamina and player 2 morale. The distance between the two fighters is also recorded.

Once the data have been collected from the bouts, operations are determined as being either single moves or combinations. While a single button press during gameplay instigates the capture and spooling of data, and each button press has a separate row, moves performed in very quick succession (within 0.2 seconds of each other) are considered to be combinations, and are identified as being single operations during the data capture phase.

Table 6.1 – Sample of data spooled from human vs. human

Time	Opponent Player			Player to be mimicked			Distance	Moves	Play
	Health	Morale	Stamina	Health	Morale	Stamina			
1.057	100	50	100	100	50	99	5.210001	j	1
1.427	100	50	100	100	50	98	5.210001	j	1
2.015	100	50	100	100	50	97	4.310001	j	1
3.123	98	50	98	98	50	95	4.850002	j	1
3.124	97	50	97	98	50	95	4.850002	*j*	1
3.159	97	50	97	97	50	94	4.850002	j	1
3.495	97	50	97	97	50	93	4.760002	j	1
3.771	96	50	97	97	50	92	4.130002	j	1
3.772	95	50	96	97	50	92	4.130002	*j*	1
4.097	95	50	96	96	50	91	4.130002	j	1
4.098	94	50	95	96	50	91	4.130002	*j*	1
4.440	94	50	95	95	50	90	4.130002	j	1
8.958	86	50	85	89	50	79	3.860002	*3*	1
20.26	79	50	54	68	50	67	4.820003	b	1
20.28	79	50	54	68	50	67	4.820003	b	1
20.32	79	50	54	68	50	67	4.820003	b	1
20.39	79	50	54	68	50	67	4.820003	b	1

Figure 6.1 below shows the high level data flow from the human vs. human bout, through to the point of execution during the human vs. AI fighter bout.

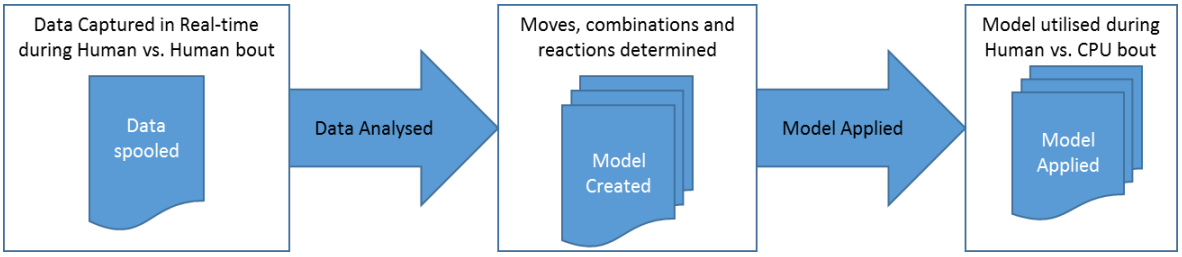

Figure 6.1 – Data Flow for KNN Solution

Figure 6.1 shows that data analysis occurs following the human vs. human bout. This also includes a record of what is perceived to be move executed as a reaction to another move. A move carried out by the player to be mimicked is considered a reaction if it is executed within an instant of the opponent carrying out a particular move. If this is repeated numerous times throughout the bout, it is considered a reaction and incorporated into a player model that is used during the human vs. AI fighter bout, such that reactions can also be emulated. Due to a threshold being set on the number of times an operation is executed in response to the opponents action before it is considered a reaction, this approach to developing a player model avoids factoring traits that could otherwise be deemed accidental or non-intentional.

6.2.3 Execution of Solution

Once the data have been captured and data analysis has been carried out to identify moves, combinations and reactions no further transformations or learning is required to take place offline ahead of the human vs. human bout. This data analysis gives rise to a model that is spooled to a single text file. The model continually monitors and accesses this file during the human vs. AI bout while the AI controls the AI player. These data are read in real-time and ultimately dictate the behavior of the AI controlled fighter. The next step during the execution phase is to calculate the Euclidean distance between the query vector ***r***, which represents real time parameters of the game at a given point in time, and each vector ***v*** in the set ***V,*** which represents the set of vectors containing game parameters collated during the human vs. human bout. Namely, the query vector ***r*** contains the six in-game parameters, player 1 health, player 1 stamina, player 1 morale, player 2 health, player 2 stamina and player 2 morale at a given point in time as a six-dimensional vector.

The all-encompassing file that is created from the human vs. human bout, containing the moves that have been performed as well as the values of the parameters under which they had been performed is referred to during the execution of the AI. This file contains all ***v*** in the set ***V***. From this point, the K nearest neighbour classifier, where K=1, is used. The values of the parameters represent the vectors belonging to ***V***, whereas the corresponding moves for each vector form the set of outputs, ***O***. The Euclidian distance between ***r*** and every element ***v*** within ***V*** is calculated using equation (5).

$$\mathrm{d}(\mathbf{r},\mathbf{v}) = \sqrt{\sum_{i=1}^{n}(\mathrm{r}_i - \mathrm{v}_i)^2} \qquad (5)$$

The vector ***v*** in ***V*** with the shortest distance to ***r*** is determined and the corresponding output from ***O*** is performed. The calculation of the Euclidean distance and selection of the output is performed during game play. As a result, any premeditated strategy that was being followed during the human vs. human bout should be replicated strictly by mimicking the operations carried out under certain circumstances dictated by the in-game parameters. It should be noted that this approach is heavily reliant on both players following the same strategy during data capture, and the human player during the human vs. AI fighter bout adhering to that strategy again. This is due to the fact that there is no framework in place that restricts the moves that can be executed as all operations fall under the same pool. As a result, if the in-game parameters do not match an example from the data collated initially during the human vs. human bout, there could be an adverse effect in the performance of the AI that manifests itself in terms of the AI fighter behavior.

Table 6.2 below contains various rows of data extracted from the file that is read by the AI during the human vs. AI fighter bout to determine which operation must be executed. The ‘moves’ column correspond with moves in Appendix 1. Data collected include the six in-game parameters, player 1 health, player 1 stamina, player 1 morale, player 2 health, player 2 stamina and player 2 morale. The distance between the two fighters, moves executed and timestamps are also recorded.

Table 6.2 – Human Bout Transcript

Time	Opponent Player			Player to be mimicked			Distance	Play through	Move
	Health	Morale	Stamina	Health	Morale	Stamina			
1.43	100	50	100	100	50	98	5.210001	1	j
2.01	100	50	100	100	50	97	4.310001	1	j
5.11	92	50	93	94	50	87	4.130002	1	j
5.94	91	50	91	92	50	85	4.130002	1	c
6.94	89	50	87	90	50	82	4.130002	1	j
7.24	88	50	87	90	50	81	4.130002	1	j
7.62	87	50	87	90	50	80	4.130002	1	j
7.98	86	50	86	89	50	79	3.500002	1	j
12.52	85	50	78	85	50	74	4.820003	1	j
13.11	84	50	76	83	50	73	4.820003	1	j
13.46	83	50	76	83	50	72	4.820003	1	j
15.99	81	50	68	77	50	70	4.820003	1	j
2.23	97	50	100	100	50	95	4.670001	3	j
2.61	96	50	100	100	50	93	4.670001	3	c
3.46	94	50	100	100	50	92	4.670001	3	j
3.83	93	50	96	98	50	90	4.670001	3	j
4.13	92	50	96	98	50	89	4.670001	3	j j
4.50	91	50	96	98	50	87	4.670001	3	j
4.80	90	50	95	97	50	86	4.670001	3	j
5.74	89	50	91	94	50	83	4.670001	3	c
9.17	81	50	81	87	50	73	4.670001	3	j
9.51	80	50	81	87	50	72	4.670001	3	j
10.36	79	50	79	85	50	70	4.670001	3	c
12.13	77	50	73	81	50	67	4.670001	3	j j
14.79	74	50	68	76	50	63	4.670001	3	j

This approach is straightforward as it uses a single AI technique, and the majority of processing is done during the execution phase, leading to few data transformations being required before a model is created. The k nearest neighbour solution is a viable solution as the strategy should be played back due to the AI executing operations in accordance to data provided, which adheres to the strategy that is being mimicked.

In instances where no particular strategy is used, and there is no qualitative means of evaluating the effectiveness of the KNN solution, the expectation is that a quantitative evaluation of the solution would yield positive results. This is due to the operations belonging to a single pool that is accessible by the AI from the beginning all the way through the duration of the bout. This non-strategic approach to playing the game is not the focus of the research carried out in this thesis, but the reality is that many players of fighting games often rely on 'button bashing', a term used for a style of play that is random and not planned in advance.

6.3 Chapter Summary

The K nearest neighbour solution presented in this chapter utilises nearest neighbour classification and focuses on playing out strategies based entirely on the operations carried out by the human player, as well as the conditions under which they are carried out. There are no restrictions in terms of the pool of moves that can be accessed by the AI at the operational layer at a given time. The theory behind this approach is that the operations would be executed by the AI under the same circumstances as they were by the human that is being mimicked. Therefore, if the human player that is being mimicked is following a pre-meditated strategy whereby moves are being executed under certain conditions, then this should be replicated by the AI.

This chapter has presented the detailed design that is largely exploiting the AI implemented at the operational level during execution. The AI mechanism has been explained, as well as the rationale and potential highlights in terms of its usage. It could be argued that there needs to be strong emphasis placed on replaying the same strategy with this approach, as this solution may be prone to error due to the fact that all operations are held centrally in a single pool, with no access control restricting when an operation is to be executed.

Chapter 7 –Data Driven Finite State Machine Solution Design

7.1 Introduction

Building on the work carried out in Chapter 6, a further solution has been designed, implemented and evaluated in this Thesis. As stated previously, the purpose of examining key techniques during the literature review in Chapter 4 was to acquire knowledge sufficient enough to create a prototype solution to pave the way in answering the aforementioned research question. Drawing inspiration from the work of Lee (2005), the DD FSM solution was split by levels of decision making; strategic, tactical and operational. As such, the solution was designed to address each level separately.

This chapter provides a detailed description of the architecture and design of the second of the two solutions to the problem of mimicking human strategies in fighting games. This solution utilises a number of existing techniques and is based on the notion of using a separate technique for each level of the decision making process. The solution itself hinges on separating pools of moves, contrary to the design of the k nearest neighbour approach, and using a data driven finite state machine to determine which pool of moves to leverage at which time. This chapter provides the detail and rationale behind the integrated solution architecture, as well as the detailed design for each component within the architecture. The architecture itself builds on the solutions presented in Saini et al (2011-1) and Saini et al (2011-2). The remainder of this chapter is structured as follows:

- Section 7.2 provides an overview of the system architecture, detailing the means in which data flows to create the overall solution that is used during a bout against the AI fighter.
- Section 7.3 provides details on how operational data are captured during the human vs. human bout.
- Section 7.4 conveys how the operational data are transformed and used to create the tactical layer of the solution.
- Section 7.5 conveys how the operations and tactical data that are captured are used to determine the overarching strategy.
- Section 7.6 establishes how the AI is executed once all the data structures have been put in place.
- Section 7.7 provides a justification for the design by means of existing research that includes unit test results for the components.

7.2 System Architecture

This section provides a high level design of the architecture for the data driven finite state machine (DD FSM) solution, as well as the rationale for the key design decisions. This section also describes how the architecture hangs together and how it was used to mimic human player strategies. A detailed account of how data are passed between core components of the architecture is covered in subsequent sections.

7.2.1 Context

Once again, the DD FSM solution is reliant on two human players playing against each other a number of times, using the same strategy each time. These data are recorded in real-time and are transcribed and saved to a file following each bout. Once the data capture phase has been completed, the data are fed into the DD FSM solution, which processes the data offline and builds the human strategy from the top down, building the strategic framework and populating it with the relevant tactics. The solution is then put into action during a bout between a human and the AI fighter, which is controlled by the DD FSM solution.

As was the case for the K nearest neighbour solution, the code for the DD FSM solution has been developed to mimic the first fighter (on the left hand side of the screen). During the execution phase, the first player shall be mimicked by the AI fighter, who shall appear on the right hand side of the screen (traditionally where the second player would be).

7.2.2 High Level Data Flow and Conceptual Architecture

Having established the difference between strategies and tactics in Chapter 4, the DD FSM solution for mimicking human strategies in fighting games is reliant upon various techniques that are instigated at various points in the decision making process. The problem is split into three tiers; operational, tactical and strategic. In the context of a one-on-one fighting game, as per the proof of concept game, an operation can be considered as an execution of a move or combination of moves; whereas a tactic can be viewed as variety of moves/combination of moves that are executed within a short space of time to achieve a particular goal. A strategy can be thought of as the bigger picture, piecing all tactics together and adhering to a set of rules under which circumstances a tactic can be executed, and which criteria must be met before deploying the particular tactic.

Following the data capture phase, all similar operations are grouped together to form a variety of groups that can be referred to as tactics. A given tactic contains similar operations (in this case, a move or combination of moves) that are alike in terms of either the circumstances under which the

operations were executed, or in terms of the timing. Details of how operations are grouped into tactics are provided in Section 7.4.

Once all tactics have been identified, the data are analysed further to determine the rules for moving from one group of operations to another. These rules are then used to create a data driven finite state machine, which forms the overarching strategy and is referred to in this research as the strategic level. Within the data driven finite state machine, the tactics/groups of operations are used to form a state, with the rules for moving from one tactic to another being defined as a state transition within the data driven finite state machine's transition table.

Figure 7.1 below shows the high level data flow from capturing the data from the initial bouts, to generating the data driven finite state machine. As shown, once the data have been captured, all operations are identified. Once this has been achieved, tactics are formed and the rules for switching between tactics are determined. This gives rise to the strategy which is implemented in the form of the data driven finite state machine. While the solution builds the data driven finite state machine from the bottom up, starting with the operational level and eventually working up to defining the strategic level, the execution of the strategy by the AI fighter is from the top-down.

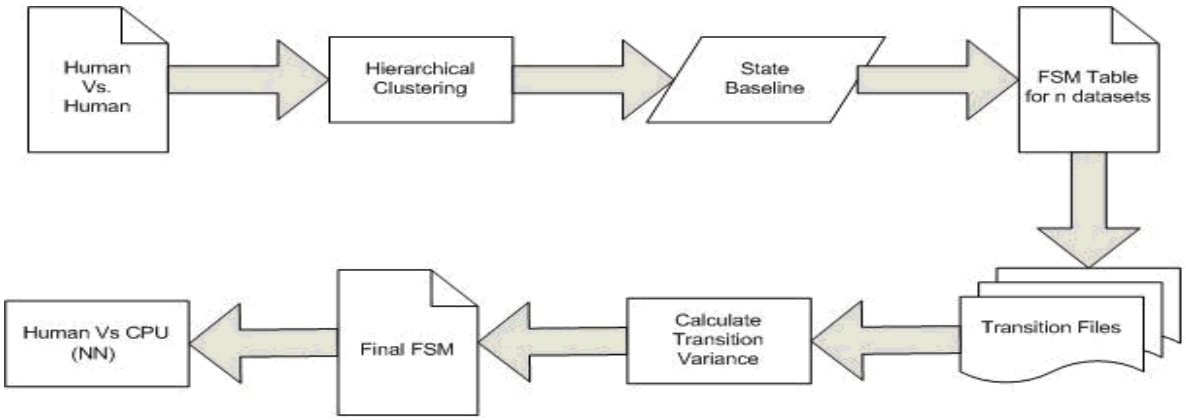

Figure 7.1 – Data flow for DD FSM Solution

The flow of data give rise to the following architecture (Figure 7.2) that creates the AI when enacted during the execute phase:

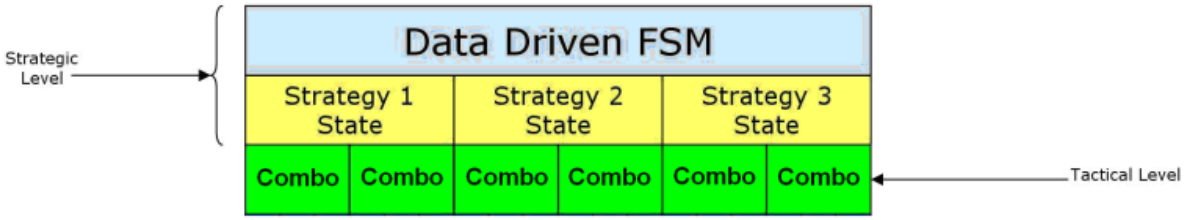

Figure 7.2 – DD FSM Solution Architecture

Architecturally, during the execution phase, the data driven finite state machine sits at the top as it is the data driven finite state machine that defines the state, and therefore the tactic to be used, which in turn limits the operations that can be executed. The in-game parameters are monitored and are constantly compared to the state transition tables that have been built. Should the in-game parameters cause a state transition, the state shall change and a separate tactic shall be used, with a different set of operations beneath it. It can be said that the execution of the DD FSM solution architecture during gameplay relies on data being fed from the top-down due to the fact that the operation is the final decision to be made and is executed immediately upon selection. The operational level itself selects an appropriate action from a pool within a current tactic/state by comparing the current in-game situation with those under which the operation executed by the human player during the data capture phase.

The DD FSM solution is designed and coded to allow data to seamlessly move between levels during the execution phase. The following sections provide further details on the design and implementation of each levels of the DD FSM solution.

7.3 Operational Data Capture

As with the K nearest neighbour solution, data capture is the initial stage of implementing the DD FSM solution. For the DD FSM solution, data is captured in much the same way as it is for the k nearest neighbour solution, by recording the data in real-time and then spooling these data to a file. Pertinent data to be spooled includes the action made by each player, and the parameters at the time of performing the action. As stated previously, data for moves performed by both players are recorded, the focus is on the first player (left-hand-side of the screen). Data that is captured takes the same format as that previously shown in Table 6.1. Once again, key data collected includes the six in-game parameters, player 1 health, player 1 stamina, player 1 morale, player 2 health, player 2 stamina and player 2 morale, as well as the distance between the two fighters. The timestamp of the moves as well as the iteration of the bout are used as the primary key for the data.

Once again, after the data have been collected from the bouts, operations are determined as being either single moves or combinations. While a single button press during gameplay instigates the capture and spooling of data, and each button press has a separate row, moves performed in very quick succession (within 0.2 seconds of each other) are considered to be combinations, and are identified as being single operations during the data capture phase. The threshold of 0.2 seconds is used this would require fast inputs on the part of the player, and makes for a convincing combination threshold akin to what is seen in martial arts. Once all operations across each play-through have been identified, the next step is to group like operations into tactics, such that each group signifies a tactic and contains a pool of operations. These groups / tactics shall eventually be used as states within the data driven finite state machine.

7.4 Generating the Tactical Level

Having established all operations related to a series of bouts that exhibit a common strategy, the next step is to create individual tactics that shall be used as states within the data driven finite state machine. There is an underlying assumption that the player being mimicked uses the same strategy each time. This section provides the detailed design for the creation of the tactical level and discusses the AI techniques used to group similar operations together.

7.4.1 Detailed Design

As stated, the overall solution addresses each of the levels of decision making with a different technique; an all-encompassing data driven finite state machine is used at the strategic level, hierarchical clustering is used at the tactical level, and K Nearest Neighbour classification is used at the operational level. Figure 7.1 shows the flow of data as well as the stages at which various techniques are used within the system architecture.

Once the operational data are collected, and before the moves within the collated data can be used to form the states for data driven finite state machine (DDFSM), they must be assigned some meaningful values such that like-instances can be grouped.

In Saini et al (2011-2), operations were assigned meaningful values and assigned to tactics. This was achieved by quantifying each of the moves and combinations of moves (herein known as a *move-set*) performed to a vector X, such that X = (x1, x2, x3, x4, x5, x6, x7, x8), where x1…x8 represent the parameters listed in Table 7.1 below.

Table 7.1 – Vector Calculation

Move	x1 No. of Moves	x2 Total Damage	x3 Damage Ratio	x4 No. of Blocks	x5 No. of Evasions	x6 No. of F.Lunges	x7 No. of B.Lunges	x8 Distance
Jab	+1	+1	x2 / x1					Distance
Cross	+1	+2	x2 / x1					Distance
Right Hook	+1	+3	x2 / x1					Distance
Left Hook	+1	+3	x2 / x1					Distance
Uppercut	+1	+4	x2 / x1					Distance
Haymaker	+1	+10	x2 / x1					Distance
Right Body	+1	+2	x2 / x1					Distance
Left Body	+1	+2	x2 / x1					Distance
Short Jab	+1	+2	x2 / x1					Distance
Short Cross	+1	+3	x2 / x1					Distance
Evade Back	+1		x2 / x1		+1			Distance
L. Evade	+1		x2 / x1		+1			Distance
R. Evade	+1		x2 / x1		+1			Distance
Push	+1	+2	x2 / x1					Distance
Block	+1		x2 / x1	+1				Distance
Low Block	+1		x2 / x1	+1				Distance
Low Kick	+1	+2	x2 / x1					Distance
Sidekick	+1	+4	x2 / x1					Distance
F Lunge	+1		x2 / x1			+1		Distance
B Lunge	+1		x2 / x1				+1	Distance

The attribute x1 represents the total number of moves executed in a given operation, while x2 represents the total damage that can be incurred by executing that operation. The damage ratio, x3, divides the total damage with the number of moves in that operation. The number of blocks are recorded in x4, evasions in x5, with front lunges and back lunges being recorded in x6 and x7 respectively. The distance between the two fighters when the operation was executed is recorded in x8. Table 7.1 lists every move in the first column, and in subsequent columns indicates what impact this move would have on the vector quantisation for x1 to x8. For example, if the move-set was a single jab executed with a distance of 4.0 between the two fighters, then X = (1, 1, 1, 0, 0, 0, 0, 4).

These vectors were then clustered using complete linkage hierarchical clustering. However, the distance threshold, beyond which clusters were not amalgamated, had to be manually edited based on the dataset. To allow for an automated approach, the move-sets are quantified based on a different vector, X = (x1, x2, x3, x4, x5) where:

x1 represents the percentage of the move-set that is based on attacks;

x2 represents the percentage of the move-set that is based on defending (blocking);

x3 represents the percentage of the move-set that is based on evasions;

x4 represents the percentage of the move-set that is based on lunges (both back and forth);

x5 represents the distance between fighters when the move-set was executed.

The clustering is achieved using the complete linkage hierarchical clustering capability found in MultiDendrograms (Fernandez and Gomez, 2008), which is used due to its capability to provide dendrogram data in text format such that it can be analysed and transformed to aid the AI solution. With the vector now quantified, a static distance criterion of 1.0 can be specified, beyond which clusters are not merged. The clustered datasets, s0, s1, s2,…., sn, act as states for a DDFSM, with moves and combinations of moves residing within each state. While the same strategy is played out three times between two human players, only data from the first bout are clustered. This creates a baseline for the number of clusters, and reduces the number of anomalies during this crucial phase of the process. Once a baseline has been created, the data from the remaining two bouts are classified against the baseline. This is achieved using K nearest neighbour classification to align each of the rows of data from the remaining bouts to a class identified during the clustering phase.

As stated, each move within the initial bout dataset is quantified as a vector $\boldsymbol{x}$, such that $\boldsymbol{x}$ = (x1, x2, x3, x4, x5) as defined above. K nearest neighbour classification is utilized by calculating the Euclidean distance between a query vector, $\boldsymbol{y}$ = (y1, y2, y3, y4, y5), representing a quantified move within the

subsequent bout dataset, and each clustered vector, ***x***. The Euclidean distance is calculated using equation (5) in much the same way it is calculated for the k nearest neighbour solution in Chapter 6. Once the Euclidean distance between a query vector, ***y***, and each clustered vector, ***x***, is calculated, the shortest distance is identified and ***y*** is classified to the corresponding state. This is repeated for each move within the subsequent bout dataset, resulting in each move over the remaining two bouts being classified to a particular state.

This approach to quantifying the vectors, ***x***, can lead to a significant amount of state transitions as move-sets are quantified without taking into account the states of neighbouring moves. For example, if the player throws a jab, then lunges back, then throws another jab, under the revised vector quantisation, this would result in two state transitions to execute these three moves. Due to this, it is highly unlikely that the strategies of each of the three human vs. human bouts shall be interpreted as being consistent with one another. To rectify this issue, and to ensure strategies are interpreted consistently, a novel algorithm has been developed and deployed in the DD FSM solution. The algorithm is executed after the clustering and classification of tactics to states and involves the following steps:

1. Data from each bout, containing which state each move-set belongs to, are listed in chronological order and state transitions are determined (not the transition functions).
2. The transition sequences from each bout are compared with one another to check if they are consistent.
3. If the transition sequences are all in line, then the algorithm terminates.
4. If the transition sequences are not in line, the bout with the longest sequence (and hence the most state transitions) is examined and the shortest state visit is identified (for example, a state may have only been visited to execute one move, then the state was exited).
5. The shortest state visit is merged with its largest neighbour. This is to say that the move-sets executed in this state for this particular transition are added to the state that was either transitioned from the state in question, or transitioned to the state in question (depending on which of these neighbours is larger in terms of how many move-sets were carried out). Following the merge, the shortest visit state is removed from the transition sequence.
6. If, following the merge, two like states are in series within the same transition sequence, these are also merged.
7. The transition sequences for each bout are re-assessed and compared with one another again to see if they are consistent. If the sequences are consistent, the algorithm terminates here.

8. If the sequences are not consistent, steps 4 – 7 are repeated until they transition sequences line up.

The above algorithm ensures that each bout's individual data driven finite state machine is consistent with that of its peers and, as such, transition functions can be determined by comparing like-for-like FSMs.

7.5 Strategic Level

This section provides details on the design and implementation of the data driven finite state machine itself to address decisions made on the strategic level of the DD FSM solution, and reproduce these decisions during a human vs. AI fighter bout.

7.5.1 Detailed Design

Having established the states, the raw data from the human vs. human bouts are re-analysed and state transitions determined. As this is a multi-parameter game, the game must be played several times between the same humans using the same strategies. Upon re-analysing the data, similar state transitions are identified. This is where the previous, current and next states for one bout are the same as those for subsequent bouts. For example, all transitions across the multiple bouts where the previous state was s0, the current state is s1 and the next state is s2, would be collated. The values of each of the six game parameters for each of the similar transitions are assessed, and the variances between the parameter values in one transition and those of similar transitions are calculated. If the variance between two of the same parameters is below a threshold of five, the parameter and its mean value is considered a transition function for that particular state transition. For example, the data shown in Table 7.2 could be considered. The threshold value of five was selected based on trial and improvement, and yielded the best results for the various strategies that were attempted. If no transition function can be found using five as the threshold, then an average of all parameters for the transition in question is taken and is used as the transition function. Having previously clustered the moves to states, once the variances have been calculated, the DDFSM can be generated. By utilizing the algorithm described, noise can be removed from the clustered dataset and a concise DDFSM can be generated without erroneous state transitions.

Table 7.2 – Like State Transitions

Prev	Curr	Next	P1H	P1S	P1M	P2H	P2S	P2M
S0	S1	S2	63	31	51	88	47	86
S0	S1	S2	61	30	40	15	34	65
S0	S1	S2	62	29	23	78	54	73
Variance			0.7	0.7	132.7	1044.3	68.7	75.0

In Table 7.2, 'Prev' represents the previous state, 'Curr' represents the current state, and 'Next' represents the next state. In the remaining columns, 'P1' and 'P2' represent player 1 and player 2 respectively, which are suffixed with 'H' for health, 'S' for stamina or 'M' for morale. For example, 'P1H' means 'Player 1s Health'.

Table 7.2 shows three similar transitions, as well as the values of each of the parameters when the transition occurred during the human vs. human bouts. The variance is calculated for each of the six game parameters. Player 1 health and player 2 stamina have a variance below the threshold; therefore, it is assumed that these parameters trigger the state transition. The mean value across the three bouts for these parameters is calculated and is used as the threshold for this particular transition function. It is deduced that when player 1's health falls below 61, and player 1's stamina falls below 30, the AI fighter can move from state s1 to state s2, provided the previous state was s0.

Taking inspiration from Lee (2005), by classifying the levels of play as either strategic or tactical, a specific AI technique can be used to tackle each level, with information being passed between levels. The strategic level is governed by a data driven finite state machine (FSM) used to model the players' various strategies and how/when the player transitions into a particular tactic. While a traditional finite state machine was previously cited as being a weak technique due to predictability and lack of flexibility at the tactical level, a data driven finite state machine at the strategic level rectifies these weaknesses. While the underpinning principal of splitting the decision making process into multiple levels is similar to Lee's (2005) architecture, the techniques used here, as well as the context of the usage varies significantly.

7.6 Execution

The design presented in the previous sections addresses each of the levels of decision making with a different technique; an all-encompassing data driven finite state machine is used at the strategic level and hierarchical clustering is used at the tactical level. The strategic level is referred to as being the

long term fighting style used throughout the bout, whereas the tactical level is concerned with short term moves and combinations that facilitate the overall strategy. Finally, the operational level, where moves are executed in real-time during the human vs. AI fighter bout, is supported by nearest neighbor classification, which essentially leverages the K nearest neighbour design documented in Chapter 6.

Figure 7.3 presents the execution process of the DD FSM solution, with example values populated to help contextulise the process. Having previously clustered the moves to states, and once the variances have been calculated, the data driven finite state machine can be generated. During gameplay against the AI fighter, once a state within the DDFSM has been entered, an operation is selected. This is achieved by calculating the Euclidean distance between the query vector ***r***, which represents real time parameters of the game at a given point in time, and each vector ***v*** in the set ***V,*** which represents the set of vectors containing game parameters collated during the human vs. human bout. The query vector ***r*** contains the six in-game parameters, player 1 health, player 1 stamina, player 1 morale, player 2 health, player 2 stamina and player 2 morale at a given point in time as a six-dimensional vector.

Each state has a corresponding file containing the moves that are to be performed as well as the values of the parameters under which they had been performed during the human vs. human bouts. The values of the parameters represent the vectors belonging to ***V***, whereas the corresponding moves for each vector form the set of outputs, $\boldsymbol{O}$. The Euclidian distance between ***r*** and every element ***v*** within ***V*** is calculated using equation (5), much like it is calculated in the k nearest neighbour solution. The vector ***v*** in ***V*** with the shortest distance to ***r*** is determined and the corresponding output ***o*** from $\boldsymbol{O}$ is performed. The calculation of the Euclidean distance and selection of the output is performed during gameplay. This entire approach of selecting the appropriate operation during the execution phase of the DD FSM solution leverages the k nearest neighbour solution in its entirety, making it a component of a much larger solution.

The proof of concept game code constantly monitors the in-game statistics during the human vs. AI fighter bout, and cross references against the data driven finite state machine that has been generated to check for any valid state transitions. If and when a transition occurs during the human vs. AI fighter bout, the state file corresponding to the current tactic shall no longer be used to select an appropriate operation. Instead, the data driven finite state machine shall mandate that the newly assigned 'current' state file be monitored, as the current tactic and an appropriate operation shall be selected from this file based on the K nearest neighbour classification.

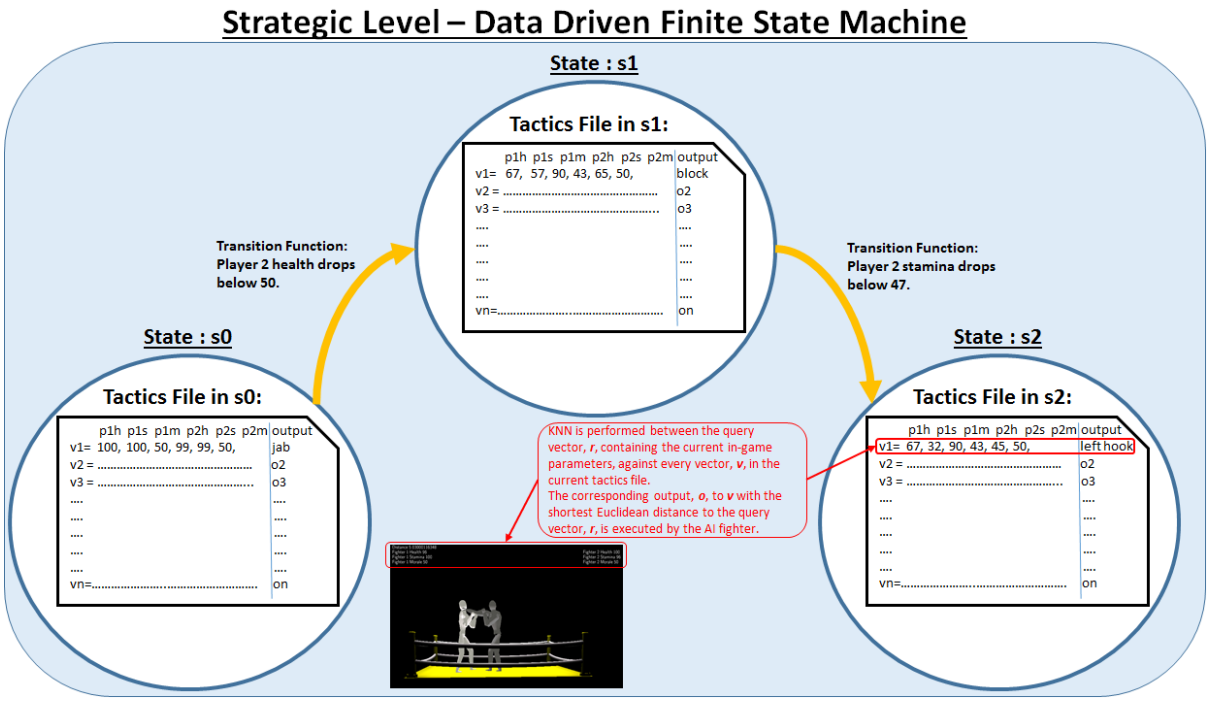

Figure 7.3 – Execution of DD FSM Solution

As shown in Figure 7.3, each tactic file forms a state within an overarching data driven finite state machine. The tactic files contain operations and are only accessed when the AI fighter is in that state. Different tactics files are accessed only when a transition occurs within the DD FSM and a new state is selected. Once within a state, the operation is selected by performing KNN classification between the query vector, ***r***, and every vector ***v***, within the current tactic file. The vector ***v*** with the shortest distance to ***r*** is selected and the corresponding operation/output, ***o***, is performed. Figure 7.3 shows three states, each with their own tactic file which contains numerous rows of data to compare the query vector against.

The example DD FSM solution presented in Figure 7.3 has three states, with the initial state being s0. When the human vs. AI fighter bout starts, the current state is set to s0 and the corresponding tactics file is read. K nearest neighbour classification is performed on in-game parameters (player 1 health, player 1 stamina etc.) using equation (5) to determine which output to select from the s0 tactics file. Once the operation is selected, the AI fighter moves into position and executes the move. The in-game parameters are read continuously and operations from the s0 tactics file are executed accordingly. In the background, the AI monitors stats to spot any state transition functions that may come to fruition. In the context of Figure 7.3, when the AI fighter's health drops below 50, a state transition occurs and the tactics file being read switches from s0 to s1. It is now the s1 tactics file that is used for the K

nearest neighbour classification for the query vector representing the current in-game parameters. Figure 7.3 shows this as being a more defensive state, including blocks as the operations that are to be executed. Once again, the AI listens for any state transition that may occur, which in this case happens when the AI fighter's stamina drops below 47, which may come about due to excessive blocking. Once this state transition takes place, the s1 tactics file is no longer read, and focus shifts to the new current state, s2. Once in s2, the in-game parameters are read as with previous states, and compared to every entry in the s2 tactics file using K nearest neighbour. As suggested by Figure 7.2, this state appears to be a move offensive state including hooks as the operations to be performed. There are no further state transitions from here, so the AI fighter remains in s2 until the game is over.

7.7 Design Justification

Once data have been collected, an experiment demonstrating a similar capability in Saini et al (2011-2) yielded largely positive results. However, this was not without its flaws. The original approach documented in this paper quantified the vector X such that X = (x1, x2, x3, x4, x5, x6, x7, x8), where x1…x8 represent the parameters previously discussed and listed in Table 7.1.

These vectors were then clustered using complete linkage hierarchical clustering using equation (3). This is where the distance between two clusters is defined as being the distance between the two furthest elements. In Saini et al (2011-2), a distance criterion of two was set, beyond which clusters are not merged. The clustered datasets, s0, s1, s2,….,sn act as states for a DDFSM, with moves and combinations of moves residing within each state.

To demonstrate the effectiveness of this approach, a strategy was formulated and played out three times in human vs. human bouts. The strategy and its associated tactics and operations are highlighted in Table 7.3 below. The 'Description' column in Table 7.3 contains short text describing the current segment of the strategy and how the player should be playing the game. The 'Moves Performed' column is self-explanatory and contains the operations that should be performed during that particular phase of the bout. Moves within the same set of square brackets are to be considered combination and should be executed in quick succession as a single operation.

Table 7.3 – Strategy for Human vs. Human Bout

Description	Moves Performed
Begin by performing long range moves/combinations at a distance.	[Jab, Cross] [Jab, Jab, Cross] [Jab] [Cross]
If health depletes below 68, block opponents attacks	[Block] [Low Block]
If stamina depletes below 55, begin evading the opponent's attacks.	[Evade Back] [Evade Left]
If player's morale exceeds 75, begin performing close range attacks.	[Uppercut, Right Body] [Uppercut] [Right Body, Left Body] [Low Kick, Left Body]

After collating the data from the three human vs. human bouts, the clustering is performed using the complete linkage hierarchical clustering capability found in MultiDendrograms (Fernandez and Gomez, 2008). The clustering gives rise to states, each containing tactics as outlined in Table 7.4 below. Hierarchical clustering was used to offer a level of flexibility and avoid the use of pre-determined states, as exhibited in Saini et al (2011-1). The 'State' column represents individual clusters, with the 'Operations' column listing the operations/moves-sets belonging to that particular cluster.

Table 7.4 – Generated States

State	Operations
s0	[Jab, Cross] [Jab, Jab, Cross] [Jab] [Cross] [Right Body, Left Body] [Uppercut]
s1	[Block] [Low Block]
s2	[Evade Back] [Evade Left]
s3	[Uppercut, Right Body] [Uppercut] [Right Body, Left Body] [Low Kick, Left Body]
s4	[Uppercut]

Once the states have been established, *like* state transitions are identified and variances between the parameters amongst the like-counterparts are calculated. The data driven finite state machine described in Table 7.5 is created and used during the human vs. AI fighter bout. The first three columns of Table 7.5 represent the previous, current and next states respectively, with the fourth column stating the transition function that must come to fruition in order for the DD FSM to move from the current state to the next state.

Table 7.5 – Data Driven Finite State Machine

Previous	Current	Next	Transition Function
null	s0	s1	AI Health < 67
s0	s1	s2	AI Stamina < 52
s1	s2	s3	AI Morale > 76

The FSM shown in Table 7.5 is in accordance to the strategy outlined in Table 7.3. When the DDFSM is actioned during gameplay, once within a state, the appropriate moves are selected. Table 7.6 contains snapshots of data at certain intervals, outlining the moves that were performed under various circumstances. The first three columns of Table 7.6 contain the human player's health, morale and stamina respectively. The next three columns contain those of the AI fighter, with the final column

detailing the moves that were carried out by the AI fighter when the aforementioned statistics in occurred. Table 7.6 shows that the moves selected at the operational level from the pool of moves within each state fall in line with the strategy outlined in Table 7.3, therefore demonstrating the usefulness of the technique.

Table 7.6 – Realtime Data Snapshots

P1 Health	P1 Morale	P1 Stam	AI Health	AI Morale	AI Stam	Moves
100	50	100	100	50	99	Jab, Jab
90	50	78	81	50	89	Cross
89	50	78	81	50	88	Cross
88	50	76	78	50	87	Jab, Cross
87	50	65	66	50	84	Block
87	50	59	66	50	78	Block
87	50	52	66	50	73	Block
87	50	46	66	50	68	L Block
87	50	37	66	50	59	Block
87	50	31	66	52	53	Back
87	50	23	66	68	53	Back
87	50	20	66	74	53	Back
83	50	19	66	76	50	Upper
55	50	19	66	76	44	Upper
51	50	19	66	76	42	R.Body L.Body
31	50	19	66	76	38	R.Body
23	50	19	66	76	36	L.Body

The results of the demonstration presented in Tables 7.4, 7.5 and 7.6 indicate that both the tactics and the overall strategy have been successfully mimicked. There are no restrictions on the number of states that can be implemented. Furthermore, this proposed architecture can cater for multi-

parameter transitions. However, there was no noise reduction or data smoothing implemented. This led to anomalies in the data caused by human error during the human vs. human bouts, which have the potential to prevent the successful application of this approach.

If a human player does not play out their strategy exactly in a number of bouts, the variance between like transitions may exceed the threshold, thus invalidating the DDFSM. Further to this, the vector calculation presented in Table 7.1 treats each value of the input vector with equal importance. In the approach represented in Saini et al (2011-2), there is no means of weighing certain elements of the vector V, to give them more importance when determining the state that particular operation should belong to. For example, Table 7.4 shows various attacks belonging to s0, including left/right body shots and uppercut, largely because these moves were performed at the same distance as the intended long range moves. These moves were not executed during the human vs. AI fighter bout as the Euclidean distance was shorter to the jab and cross moves, however, they should not belong to s0. There is potential to rectify this by assigning weights to each value of the vectors.

It could also be argued that this technique is not entirely automated due to the human intervention in selecting a sensible distance criteria beyond which clusters are not merged. Additional enhancements were required to smooth the data and to ensure the strategy has been interpreted correctly.

To rectify this, a key design decision is made with regards to the tactical level, and generating the states, within which the operations shall reside, pertaining to the vector quantisation of the operations. The design proposed in this chapter has the operations being quantified to a vector X = (x1, x2, x3, x4, x5) as described in Section 7.4.1.

The design documented in this chapter follows lessons learned from the research documented in Saini et al (2011-2), as the revised 5-dimensional vector quantisation provides a more meaningful representation of the operations, as well as enabling the distance criterion of the hierarchical clustering to be fixed at 1.0, while providing consistent results. To aid this enhancement and to ensure there was a level of data smoothing, such that the number of states would not make for un-implementable finite state machines, the algorithm described in Section 7.4.1 was developed.

7.8 Chapter Summary

This chapter has presented the high level architecture for the second of the two proposed solutions for the problem of mimicking human player strategies in fighting games. The DD FSM solution is reliant on a number of existing AI techniques including data driven finite state machines, K nearest neighbour

classification and hierarchical clustering. The usage of the different AI techniques accommodate the approach of splitting strategies and tactics, with a different technique addressing each level of the decision making process. Details of a novel algorithm that has been developed to aid the architecture have also been provided. The novel solution presented in this chapter also differentiates between capturing the data, interpreting and subsequently identifying strategies and tactics, before finally executing the strategy. A justification for the architecture has been provided.

Having established the design for a solution to the problem pertaining to mimicking human strategies in fighting games, the solution can now be implemented. Following implementation, the solution can be evaluated using the proof of concept game described in Chapter 5 as a test bed. A suitable means of evaluating that this solution is fit for purpose must be established. The design and execution of such an evaluation in covered in Part III of this Thesis.

Part III – Evaluation and Conclusion

Chapter 8 - Evaluation

8.1 Introduction

Up to this point, the problem domain related to mimicking human strategies in fighting games has been defined, as well as two proposed solutions for solving this problem. A detailed design has been provided for each solution, as well as a justification for the base architectures and implementation methods. An evaluation study has been carried out to evaluate the two proposed solutions. This determines whether the solutions are fit for purpose, as well as identifying whether or not one solution performs better than the other. This chapter provides details how each of the proposed solutions are to be evaluated. Each solution is subject to the same experiment, with the same set of test data. The experiment design is described as well as the strategies that drive the data being fed into each solution. This chapter also provides the results as well as the analysis and interpretation of the results.

It was important that both solutions were evaluated thoroughly. To this end, both qualitative and quantitative measures were taken by having expert gamers observe and evaluate the effectiveness of the techniques, as well as capturing transcripts of the bouts in question. Both solutions were evaluated in the same manner, providing a comparison of multiple techniques.

Both techniques are evaluated by means of an experiment, which has been designed to conduct both a quantitative and qualitative analysis of each of the techniques. A qualitative measure of how human perceives the AI fighter to be performing is taken, as well as transcripts of moves carried out during each bout between humans and corresponding bouts between a given human player and the AI fighter.

8.2 Experiment Design

Having considered the success criteria, as well as the aims and objectives of the research, an experiment was designed and carried out to evaluate both the k nearest neighbour and data driven finite state machine (DD FSM) solutions. Special considerations were given to the success criteria for the solutions being evaluated. Quantitative and qualitative measurements were used based on the reasons given below.

A qualitative measure is required to understand the perceived performance of the AI, as ultimately, in a real world scenario when playing any videogame, it is a human player who must evaluate how the AI is performing. Although, in the real world, this is typically against a criteria of whether or not the AI

is good or is playing sensibly. In the context of this research, this criteria is adjusted such that a human is taking a qualitative measure on how the AI is performing in terms of playing out the strategy of a given human player. An additional criteria to whether the AI is being perceived to play out a particular strategy, is whether the AI is actually 'behaving' as a human would. Therefore, an assessment must be made by a human as to the closeness of the AI mimicking behaviour. A quantitative measure is required to either support, or refute the finding of the qualitative evaluation. This is an important measure as it quantifies the usefulness of the AI, but also enables a view on the qualitative evaluation in terms of either lending credibility to the perception of the human subjects, or by suggesting that perception of the AI performance is a far cry from the reality. Combining these two types of evaluation and generating results from the same source data and experiment is critical to understanding whether either, or both of the AI solutions are fit for purpose, or if one is superior to the other in particular, if not all circumstances.

This is achieved by first having two human players play against one another in the proof of concept game described in Chapter 5, with the first player adopting a pre-defined strategy, while the second player is left to play the game as he/she chooses, but must play this way with consistency for each bout where their opponent is leveraging a particular strategy. Both players are to engage in combat using the PlayStation 3 Sixaxis control pad. The pre-defined strategy is played by the same player three times, i.e. over three separate bouts. Each pre-defined strategy is fully known to the first player who is responsible for acting out the strategy. The high level strategy is summarised, as well as pivotal moments when the tactics must change. These ultimately involve monitoring all in game statistics and knowing when to change tactics, as well as the operations to use when executing a particular tactic.

There are a total of ten pre-defined strategies outlined in Table 8.3, that are played out three times each, making for a total of thirty bouts. It is imperative that each strategy is played out accurately by both players in each bout. Data collated from bouts containing an abundance of human generated errors due to the strategy not being played out properly shall be expunged and relevant human vs. human bout shall be played out again. Each of the thirty bouts are individually video captured for reference as these videos shall play a large part in the qualitative evaluation. Following these bouts during the data capture phase, the data generated for each bout of a given strategy are used to build both a K nearest neighbour and DD FSM model. This is repeated for each strategy, however the data are segregated so as to ensure each AI is only representing a single strategy. In the case of the DD FSM approach, when the model is being generated as described in Chapter 7, the data driven finite state machine is defined, as well as the states which represent tactics. The DDFSMs are recorded in Table

8.4. Once the DD FSM solution and KNN solution have generated the models for each of the ten strategies, the human vs. AI fighter bouts can commence.

The human vs. AI fighter bouts for both techniques are video captured and transcripts of the AI fighter's actions are recorded. This makes for two additional videos per strategy as each solution is captured in its own video. The transcripts are generated in much the same way as during the data capture phase, with each entry having a timestamp as well as each of the six in game parameters; player 1 health, player 1 stamina, player 1 morale, player 2 health, player 2 stamina and player 2 morale. The distance between the two fighters is also recorded. As during the data capture phase, data are only captured and spooled when an operation is executed. These transcripts, along with the data driven finite state machine definitions created during pre-processing, shall form the foundation of the quantitative analysis. Table 8.1 and Table 8.2 below contain an extract of data taken from such transcripts of the DD FSM solution and KNN solution respectfully.

Table 8.1 – DD FSM Solution Transcript

Opponent Player			AI Player			Distance	State	AI Player Moves
Health	Morale	Stamina	Health	Morale	Stamina			
100	50	100	100	50	100	5.660001	s2	_ u_
100	50	98	96	50	100	4.490001	s2	_ j j_
100	50	96	94	50	100	4.490001	s2	_ c j_
100	50	93	91	50	100	5.300004	s2	_ j c_
99	50	90	88	50	98	4.850004	s2	_ j_
98	50	89	87	50	97	4.850004	s2	_ j c_
82	50	79	77	50	82	4.670002	s2	_ j j_
80	50	77	75	50	79	4.670002	s2	_ j j_
65	50	63	59	50	63	3.590002	s2	_ j j_
64	50	57	51	50	62	4.490002	s2	_ j_
59	50	56	50	50	58	4.490002	s1	_ u_
58	50	48	46	50	51	4.670002	s1	_ b b b b_
58	50	44	46	50	47	4.670002	s2	_ b b b_
58	50	41	46	50	45	4.490002	s2	_ 1 1_
57	50	39	42	50	42	3.680002	s2	_ 1_
47	50	36	36	50	34	3.410002	s2	_ 1_
47	50	35	34	50	29	4.220003	s2	_ 1_
33	50	34	32	50	20	4.760001	s2	_ 1 j u_

Table 8.2 – KNN Solution Transcript

Opponent Player			AI Player			Distance	Moves
Health	Morale	Stamina	Health	Morale	Stamina		
100	50	100	100	50	100	8	__
100	50	100	100	50	100	5.570001	_ u_
100	50	98	96	50	100	4.580001	_ j j_
100	50	96	94	50	100	4.580001	_ j c_
100	50	93	91	50	100	4.580001	_ j_
100	50	92	90	50	100	4.580001	_ c j_
100	50	89	87	50	99	5.030005	_ j j_
97	50	87	85	50	93	4.490004	_ j_
74	50	86	84	50	77	5.210003	_ j_
73	50	85	83	50	75	5.660001	_ j j_
70	50	83	81	50	70	4.760001	_ j_
67	50	82	80	50	62	4.130001	_ c_
64	50	80	78	50	58	4.580002	_ b_
64	50	80	78	50	56	5.570002	_ c c_
62	50	76	74	50	51	4.490002	_ b b b b b b_
61	50	70	74	50	45	4.580002	_ j b_
60	50	67	73	50	44	4.580002	_ b b b b b b_
60	50	61	73	50	40	4.580002	_ b b b b b b b b_
60	50	57	73	50	36	4.580002	_ b b b b b b_
60	50	53	73	50	33	4.580002	_ u_
58	50	50	69	50	32	4.130001	_ u u_
58	50	46	61	50	30	3.590001	_ 1 1_

Both the human vs. human, and the human vs. AI fighter sets of videos that are captured play a pivotal part of the evaluation of each of the techniques as they are viewed by ten individuals, referred to as observers. Each of the observers is familiar with fighting games in general, and the mechanics of the proof of concept game created for this research. They have spent time learning the rules of the game, and understanding the possible strategies that could be at the players' disposal given the arsenal of moves and actions available. The group of observers is made up of avid gamers that spend time playing commercial fighting games, thus giving them an appreciation for the proof of concept game, as well

as the overall aim of the experiment. As preparation for this experiment, each observer is shown a video of a human vs. human bout for each strategy, and is provided with the strategy breakdowns in Table 8.3. This is to give them a feel for how a known strategy is played out between humans, as well as enhancing their ability to notice changes in tactics and identifying particular actions based on their animations.

Having familiarised each observer in exactly what the ten strategies are (see Table 8.3), as well as their appearance and animations when played out by a human playing against another human, the next step is to assess the AI solutions in a qualitative manner. To achieve this, each observer is shown a further four videos per strategy. It should be noted that from this stage of the experiment onwards, the observers are to partake in the experiment in isolation from the other observers so as to rule out any influential factors. For each of the four videos, the observers are asked whether or not the strategy in the video is representative of the documented pre-defined strategy in question. This fulfils the criteria of performing a qualitative analysis on whether or not the AI has indeed accurately executed the strategy in question. The observer is also asked to identify whether the strategy being played out in the video is being executed by the human or the AI player. This fulfils the criteria of performing a qualitative analysis on whether the AI can execute the strategy without making it obvious that it is an AI player. In reality, the four videos per strategy comprise of

- Two videos of the human vs. human bouts
- One video of the human vs. DD FSM AI bout
- One video of human vs. KNN AI bout

It should be noted that the observer is tasked with determining which of the on-screen fighters is playing out the strategy before coming to any conclusions.

Table 8.3 outlines the 10 strategies that are to be played out and observed. The first column identifies that strategy number, which is a unique identifier. The second column, distance, states roughly how far the fighters should be from one another during a particular phase of the strategy. ‘Close’ suggests the fighters should be right next to each other, medium suggests that the fighters should be up to, but no more than a quarter screen away from each other, and anything beyond that is deemed far. The moves column states which move-sets / operations should be carried out during that phase of the strategy and the transition column states the parameter transition that must come to fruition before moving to the next phase of the same strategy.

Table 8.4 details the finite state machines that were generated by the DD FSM solution. Each strategy number listed in column one corresponds to those in Table 8.3. Columns two to five in Table 8.4

contain information on the FSMs including the previous, current and next state, as well as the transition function to get from the current state to the next. The remaining columns list the move-sets / operations that are executed in each state of the given strategy.

Table 8.3 – Human Strategies Outline

Strategy	Distance	Moves	Transition
1	Medium	Jab	P1 Health<70
		Jab Jab	
		Jab Cross	
	Close	Block	P1 Stam<50
	Close	R Body	N/A
		L Body	
		Short Jab, Short Cross	
2	Close	Evasions	P1 Moral>75
	Close	Haymaker	N/A
3	Medium	Hooks	P1 Stam<70
		Jab Jab Cross	
		Jab Cross	
	Far	Side Kick	P1 Health<20
	Close	Block	N/A
4	Medium	Lunge Forward, Lunge Back, Jab	P1 Stam<50
	Close	Body Shots, Uppercut	P1 Health<20
	Close	Block	N/A
5	Medium	Evasions, Jab, Cross	P2 Health < 80
	Close	Uppercut, Body Shots	P1 Health<60
	Close	Block	N/A
6	Medium	Hooks	P2 Health < 50
	Medium	Haymaker, Side Kick,, Cross	N/A
7	Medium	Jab Cross, L/R Hooks	P1 Health < 60
	Medium	Evasions	P1 Morale > 75
	Medium	Haymaker	N/A
8	Medium	Lunges, Jab	P1 Stamina <60
	Far/Medium	Side Kick, Evasion	P1 Health < 50
	Medium	Cross, Jab	P1 Health < 20
	Close	Block	N/A
9	Varies	Random	N/A
10	Varies	Random	N/A

Table 8.4 – Data Driven Finite State Machines

Strategy	Previous	Current	Next	Transition	State 1	State 2	State 3
1	Null	State 2	State 1	Human Stamina below 56. AI Health below 68.	Block	Jabs, Cross.	
	State 2	State 1	State 2	AI Health below 66. AI Stamina below 45.		Uppercut, short jab, short cross, left body blow, right body blow.	
2	Null	State 1	State 2	AI Morale above 82.	Back evasion.	Haymaker	
3	Null	State 1	State 2	AI Health below 22.	Jab, Cross. Left hook, Right Hook, Side Kick	Block	
4	Null	State 2	State 1	AI Health below 18.	Block, Low Block, Jab	Back/Forward Lunges. Jab, Uppercut, Left/Right body blow.	
5	Null	State 1	State 2	AI Health below 96.	Jab, Cross, Uppercut, Left/Right Body Blow	Evasions	Block
	State 1	State 2	State 1	Averages taken			
	State 2	State 1	State 2	Averages taken			
	State 1	State 2	State 3	AI Health below 58.			
6	Null	State 1	N/A	N/A	Cross, Jab, Left/Right Hooks, side kick, haymaker		

Strategy	Previous	Current	Next	Transition	State 1	State 2	State 3
7	Null	State 1	State 2	AI Health below 60. AI Stamina below 72.	Jab, cross, left/right hooks.		
	State 1	State 2	State 1	AI Health below 58. AI Morale above 76. AI Stamina below 67.	Haymaker	Evasions	
8	Null	State 2	State 1	Human Stamina below 96. AI Health below 96.	Evasions Side Kick	Jab, Cross, Side Kick, Lunges	Block Low Block
	State 2	State 1	State 2	Averages taken.			
	State 1	State 2	State 3	AI Health below 19.			
9	Null	State 2	N/A	N/A	N/A	Misc.	
10	Null	State 2	N/A	N/A	N/A	Misc.	

8.3 Experiment Results

The results for the observers' perception are recorded in Table 8.5. Each observed video is attributed one of the following:

- H, signifying the observer has deemed the strategy to have been executed by a human player.
- AI, signifying the observer has deemed the strategy to have been executed by an AI player.
- NA, signifying the observer has deemed the strategy in the video to be non-representative of the strategy in question.

These results are collated further in Figure 8.1, which shows the split between AI, H and NA for the DD FSM solution. This information is also shown for the K nearest neighbour solution in Figure 8.2. Figure 8.3 shows the overall picture, combining data from Figure 8.1 and Figure 8.2.

Table 8.5 – Observations

		Observers									
Video	**Video Description**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	Strategy 1 Human 1	H	H	H	H	H	H	H	H	H	AI
2	Strategy 1 Human 2	H	H	H	H	H	H	H	H	AI	H
3	Strategy 1 DDFSM	NA	H	NA	H	H	AI	AI	AI	H	AI
4	Strategy 1 KNN	AI	AI	H	H	H	AI	H	H	H	H
5	Strategy 2 Human 1	H	H	H	H	H	H	H	H	H	H
6	Strategy 2 Human 2	H	H	H	H	H	H	H	H	H	H
7	Strategy 2 DDFSM	AI	H	H	H	H	AI	H	H	H	AI
8	Strategy 2 KNN	AI	H	H	H	H	H	AI	H	H	NA
9	Strategy 3 Human 1	H	H	H	H	H	H	H	H	H	H
10	Strategy 3 Human 2	H	H	H	H	H	H	H	H	H	H
11	Strategy 3 DDFSM	H	H	H	H	H	AI	H	AI	H	H
12	Strategy 3 KNN	NA	H	NA	H	AI	AI	H	AI	NA	H
13	Strategy 4 Human 1	H	H	H	H	H	H	H	H	H	H
14	Strategy 4 Human 2	H	H	H	H	H	H	H	H	H	H
15	Strategy 4 DDFSM	H	AI	H	H	H	H	AI	H	H	AI
16	Strategy 4 KNN	H	NA	NA	H	AI	NA	AI	H	AI	H
17	Strategy 5 Human 1	H	H	AI	H	H	H	H	H	H	H
18	Strategy 5 Human 2	H	H	H	H	H	H	H	H	H	H
19	Strategy 5 DDFSM	NA	NA	NA	NA	NA	NA	AI	AI	NA	NA
20	Strategy 5 KNN	AI	H	NA	AI	AI	NA	H	H	AI	NA
21	Strategy 6 Human 1	H	H	H	H	H	H	H	H	H	H
22	Strategy 6 Human 2	H	H	H	H	H	H	H	H	H	H
23	Strategy 6 DDFSM	AI	AI	AI	H	H	H	H	AI	AI	H
24	Strategy 6 KNN	AI	AI	H	H	H	H	H	AI	H	H
25	Strategy 7 Human 1	H	H	H	H	H	H	H	H	H	H
26	Strategy 7 Human 2	H	H	H	H	H	H	H	H	H	H
27	Strategy 7 DDFSM	AI	H	H	NA	AI	H	NA	NA	H	H
28	Strategy 7 KNN	AI	AI	NA	AI	NA	AI	H	NA	H	H
29	Strategy 8 Human 1	H	H	H	H	H	H	AI	H	AI	H
30	Strategy 8 Human 2	H	H	AI	H	H	H	AI	H	H	AI
31	Strategy 8 DDFSM	H	NA	NA	NA	AI	AI	H	NA	AI	AI
32	Strategy 8 KNN	AI	AI	AI	NA	AI	NA	AI	NA	N/A	AI
33	Strategy 9 Human 1	H	H	H	H	H	H	H	H	H	H

		Observers									
Video	**Video Description**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
34	Strategy 9 Human 2	H	H	H	H	H	H	H	H	H	H
35	Strategy 9 DDFSM	H	H	H	AI	H	AI	H	H	AI	H
36	Strategy 9 KNN	H	AI	H	H	H	H	H	AI	H	H
37	Strategy 10 Human 1	H	H	H	H	H	H	H	H	H	H
38	Strategy 10 Human 2	H	H	H	AI	H	AI	H	H	AI	H
39	Strategy 10 DDFSM	H	H	H	H	H	AI	H	H	H	H
40	Strategy 10 KNN	AI	AI	H	H	H	AI	H	AI	H	H

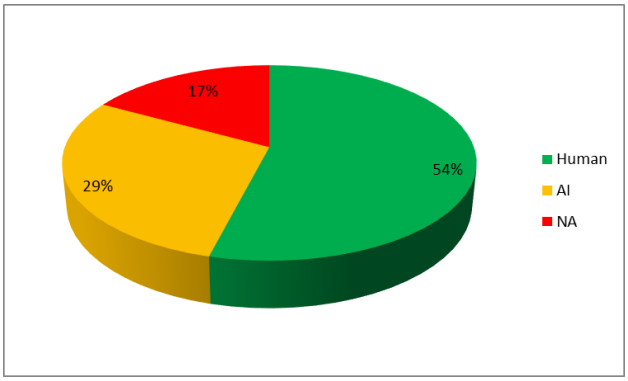

Figure 8.1 – Observer Perception of DD FSM Performance

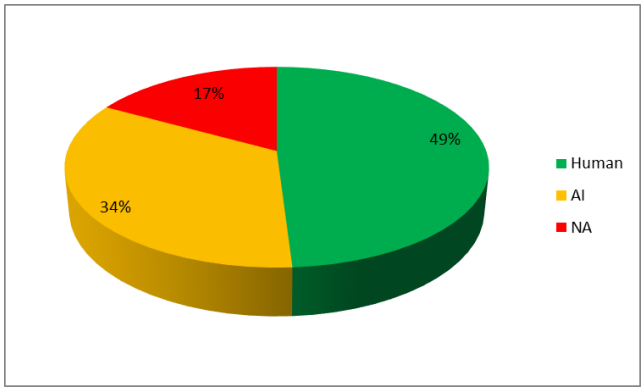

Figure 8.2 – Observer Perception of KNN Performance

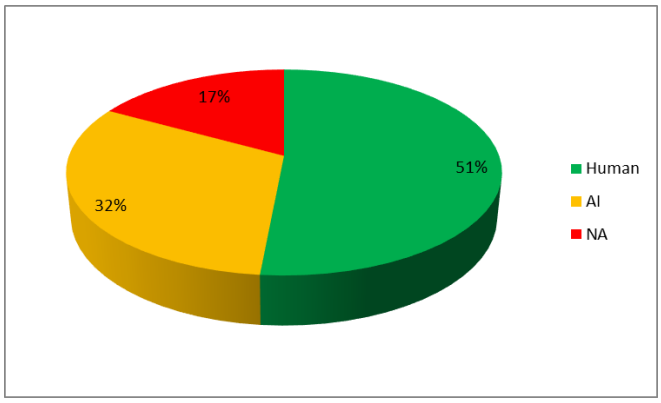

Figure 8.3 – Observer Perception of Both Solutions Combined

Figure 8.1 shows that the DD FSM solution produced videos that were largely interpreted as exhibiting strategies played by a human player, with 54% of interpretations being categorised as such. Further to this, 29% of the videos were interpreted as correctly mimicking the strategy in question, although notably considered to be played out by the AI player. The DD FSM solution also yielded 17% of observations that were considered to incorrectly mimic the strategy in question. Table 8.5 shows that these interpretations are mostly of strategies 5, 7 and 8. Figure 8.2 shows that the KNN solution yielded similar results to the DD FSM solution, albeit with slightly more videos being interpreted as AI driven. Of the 100 observations, 49 considered the video to exhibit strategies played out by a human player, 34 were considered to be played out by an AI player, and 17 were considered to inaccurately mimic the strategy. The results presented in Figure 8.1 and Figure 8.2, as well as the breakdown in Table 8.5 suggest that the strategies have largely been successfully mimicked, with only 17% of the observations in each case being considered to have been non-representative of their corresponding strategy.

A closer look at the results reveals that this 17% is not common across both techniques, with the DD FSM solution categorically failing to reproduce strategy 5 convincingly. Based on notes collated by the observers, and a closer look at the FSM for this strategy in Table 8.4, it is apparent that there are more state transitions factored into the execution of this strategy than are needed. This could be attributed to poor source data gathered during the human vs. human bouts. Further to this, by its own nature, the strategy lends itself to rapid state transitions between executing lunges and throwing attacks. If this is not done with consistent timing and planning across all three human vs. human bouts, then states shall not be merged as expected.

Further ‘NA’ results for the DD FSM solution also include strategies 7 and 8, however, this is consistent with the results for the KNN solution. Observers commented on the DD FSMs use of strategy 7 showed

premature throwing of a single punch while the fighter should have been evading, as well as the fact that evasions did not start promptly following the AI fighter's health dropping below 60. The throwing of the punch is most likely an inadvertent reaction driven data collected during the human vs. human bout. Strategy 8 is a complex strategy combining lunges, evasions, attacks and blocks. The FSM for this strategy does not accurately represent the flow through states as intended when the strategy had been documented. While the strategy is largely met, because of the fact that averages were taken to transition from state 1 to state 2, this violates the documented strategy and the AI fighter enters state 2 and begins jabbing before the documented strategy would have them do. The fact that averages are taken that are inconsistent with the documented strategy suggests that the source data were not as clean and consistent as expected.

Generally speaking, it is apparent that KNN solution generated AI fighters that were considered not to have accurately mimicked the strategy due to prematurely executing moves ahead of the threshold parameters being met. For example, in strategy 8, the AI fighter being controlled by the KNN AI begins evading too early, and mixes jabs and crosses with the evasions, hence violating the strategy. Another example can be found during strategy 4 where the fighter blocks some time before its health is depleted beyond 20. This strongly suggests that the KNN solution suffers as there are no structures to conform to, like those provided by FSMs.

However, the rigidity offered by the DD FSM solution often causes additional parameters to be considered part of the transition functions, as evidenced in Table 8.4. The first transition in strategy 1 requires the player's health to have depleted by 70, and nothing more. However, based on the data collated during the human vs. human bouts, the DD FSM considers the opponent's stamina to drop below 56 as a further requirement. This does not cause a great issue, as in most cases, these residual transition functions are met by default, especially if the human opponent is playing the AI in such a way that the bout between the two humans is replicated.

The DD FSM solution does offer a level of flexibility to the degree that there may only be one single state from which the operations are selected, which is exactly how the KNN solution works. For the solution to create such a DDFSM, the data across all human vs. human bouts must be sporadic as if no strategy is being used. This was the approach for strategies 9 and 10. Ultimately, the AI model for both DD FSM and KNN solutions were identical in that both architectures had a single pool of operations and tactics and there was no structure in place safe-guarding certain move-sets. This was also the case for strategy 6. However, in this instance it was due to the fact that all move-sets had been clustered to the same state. In the cases of strategies 9 and 10, the focus was on replicating the sporadic, reactive nature of the fighter being mimicked. Judging by the results, this was successful.

Based on discussions with the observers, the differentiation between interpreting the observed strategy as being carried out by either human or AI is largely based on the observer's impression of the fighter executing the strategy. There were instances during the human vs. AI fighter videos where punches were thrown just out of striking distance, or lunges were made with minor glitches, suggesting immediately to the observer that they are indeed watching an AI mimic a human strategy. While the observers are experienced gamers with a keen interest in fighting games, such glitches are noticed easily, and would most likely be detected by even a novice gamer watching a fighting game being played due to the graphical representation of the on-screen fighters. This by no means suggests failure as the scope of this research is to evaluate techniques to mimic human strategies, rather than evaluating AI techniques to play the game such that observers perceive the game to be played by a human. If the latter was the objective of this research, then one would have to consider introducing randomness and deliberate errors.

The research carried out in this chapter suggests that both the DD FSM and KNN solutions are capable of mimicking human strategies. Figure 8.3 shows that 83% of observations of the AI techniques considered the strategies to have been successfully mimicked. Combined and individually, both solutions have surpassed the success criteria which was set at 75% (see Chapter 5) by achieving an 83% success rate.

While both techniques are equally matched, with each only having 17% of negative observations where it was deemed the AI did not mimic the strategy accurately, it is arguably the DD FSM solution that is the stronger of the two. This is due to the fact that the 8% of the 17% of negative observations could be attributed to poor raw data for strategy 5. Further to this, the DD FSM solution also generated the greater number of videos that were observed to have been played out by humans.

8.4 Chapter Summary

This chapter has provided details pertaining to the means of evaluating both the DD FSM solution and the KNN solution. The approach has been to use both qualitative and quantitative analysis to evaluate the solutions. A set criteria for the qualitative analysis has been presented, which call for a human interpretation as to (i) the solutions are indeed mimicking the strategy in question accurately, and (ii) the solutions are mimicking the strategy in such a way that it can be perceived to be a human playing the strategy rather than the AI fighter. The experiment design and results have been presented and a deep analysis has been conducted, which suggests that both solutions perform equally well but the scoring varies across the strategies. This can be attributed to the advantages and disadvantages of each solution.

The results yielded by the experiment carried out on each technique can be interpreted and summarised as follows:

- Of the videos featuring the DD FSM solution mimicking human strategies, 54% were observed and interpreted as being the correct strategy being played out by a human. 29% were observed and interpreted as being the correct strategy, albeit played out by an AI, and 17% were observed and interpreted as incorrectly mimicking the strategy in question.
- The majority of strategies that were viewed as being incorrectly mimicked by the DD FSM solution were strategies that featured rapid state transition, and where the source data is not as clean as it could be.
- In some instances, the DD FSM solution over complicates the transition function by capturing triggers from other parameters and factoring them into the state transition function. For example, there are instances where the only trigger to move from one state to another is the health dropping below a certain threshold. However, the DD FSM solution may also consider stamina as being a pertinent parameter due to the averages used to generate the data driven finite state machine.
- The KNN approach yielded similar results to the DD FSM approach, albeit with slightly more videos being interpreted as AI driven. Of the KNN solution observations, 49% were interpreted as being the correct strategy being played out by a human. 34% were observed and interpreted as being the correct strategy, albeit played out by an AI, and 17% were observed and interpreted as incorrectly mimicking the strategy in question.
- The strategies that were incorrectly mimicked by the KNN solution are more sporadic, and are the result of the AI prematurely executing moves ahead of threshold parameters being met.
- The research carried out in this chapter suggests that both the DD FSM and KNN solutions are capable of mimicking human strategies as 83% of observations of the AI techniques considered the strategies to have been successfully mimicked.
- While both techniques are equally matched, with each only having 17% of negative observations where it was deemed the AI did not mimic the strategy accurately, it is arguably the DD FSM technique that is the stronger of the two. This is due to the fact that the 8% of the 17% of negative observations could be attributed to poor raw data for strategy 5. Further to this, the DD FSM technique also generated the greater number of videos that were observed to have been played out by humans.
- The success rate of 83% achieved by both solutions surpasses the success criteria which was set at 75%.

Chapter 9 – Conclusion and Future Work

9.1 Introduction

This chapter provides the final summary of the Thesis and draws together the main conclusions of the research. An overall review of the Thesis and the overall conclusions that have been drawn is provided, as well as how the main conclusions link back to the aims and objectives. A summary of contributions is also provided, and limitations of the research carried out, both in terms of the solutions developed and the research as a whole, are discussed. This chapter also identifies future work that can be carried out, building on the research documented in this Thesis.

9.2 Thesis Review

The original aim of this research was to answer the question; 'How can existing Game Artificial Intelligence techniques be used to successfully mimic basic human player strategies in strategic fighting games?' The aim was to answer this question by researching, identifying, implementing and evaluating an AI system capable of mimicking basic strategies of human players in fighting videogames. The literature review presented in Part 2 of this Thesis has further validated the void in current research and justified the aim of this work. Prior to this research, there was no research conducted in the field of game AI pertaining to mimicking strategies and tactics in fighting games. This Thesis has consolidated research around the problem domain and provided detailed background on the context of the problem, as well as detailing the nature of fighting games.

The research presented here has gone a long way to answering the original research question by designing, developing and implementing and evaluating two separate and novel techniques that can be used to mimic basic human player strategies in strategic fighting games. The detailed designs presented in Chapters 6 and 7 were implemented by means of a unique proof of concept game that was developed specifically for this research, as detailed in Chapter 5. Following the design and build of the proof of concept game, as well as the AI solutions, an evaluation method was identified and executed in Chapter 8, which yielded positive results. The research within this Thesis is builds on the work presented in Saini et al (2011-1) and Saini et al (2011-2).

In summary, various architectures have been discussed, ultimately being refined to two; one DD FSM approach that has been developed and refined throughout the lifecycle of this project, and one K nearest-neighbour classification based solution. In the context of the proof of concept game developed, both solutions have succeeded in mimicking player strategies to a large extent. However, the K nearest neighbour solution suffers from poor accuracy on occasion. In a real world scenario,

where players may not necessarily play strategically with a pre-determined strategy in mind, the k nearest neighbour solution may hold an advantage in that it is capable of replicating operational behaviour synonymous with so-called 'button bashers'. As the results in Chapter 8 suggest, such a scenario would likely give rise to performance issues for the DD FSM solution due to the excessive number of state transitions.

Provided the source data is sound from the human vs. human bout, such that they are not interpreted as triggering an overly excessive number of state transitions, the DD FSM solution performs well by providing a framework, forcing the AI fighter to behave accurately in accordance to the input data.

9.3 Summary of Contributions

The major contributions made by this Thesis can be summarised as the following:

- The DD FSM solution has been developed, which constitutes a novel AI solution architecture based on amalgamating various AI techniques. The solution is novel in that the context in which the techniques are used, and the manner in which the design calls for these specific techniques to address each level of the decision making process, has not been implemented and evaluated in the fighting game previously. The design of the architecture itself is original, as is its usage to mimic human strategies in fighting games, or any other game genre. Research in the field of Game AI has not yielded such a solution, nor any similar solution and applied it to the novel problem domain of mimicking human strategies in fighting games.
- As part of the DD FSM solution, a novel algorithm to ensure data smoothing amongst state transitions for data driven finite state machines has been developed. The steps implemented to smooth the data are unique and have not been applied to a problem of this nature previously. This algorithm has the potential to be implemented in similar problems to define a data driven finite state machine.
- The K nearest neighbour solution has been developed, constituting a bottom-up AI architecture for mimicking human strategies in fighting games. This usage of the k nearest neighbour classification technique is novel with respect to its application in a strategic fighting game. The approach to mimic and execute a fighting game strategy by only identifying the operations and building the strategy from the bottom up has not been covered in previous research. This is a novel design, the implementation of which has been documented for the first time as part of this research.
- The research project has also yielded specific knowledge in the field of AI and its application to fighting games in particular. The results of the experiments in Chapter 8 indicate that both

solutions can be used to solve the problem pertaining to mimicking human strategies in fighting games. The knowledge that has come to light following the experiments in Chapter 8 affirms that both solutions are fit for purpose, and the respective designs can indeed yield positive results when implemented in a strategic fighting game. The results of the experiments also suggest that the DD FSM solution is better at mimicking more methodical, structured strategies. These are strategies where there is no deviation in the usage of tactics as bouts progress, and the move-sets that are used are not too sporadic, for example, lunges and blocks are not randomly interspersed with offensive moves. This is in contrast to the k nearest neighbour solution, which is better at mimicking strategies that may be more fluid in that the source data may not always be consistent. However, the k nearest neighbour solution does not provide the high level of accuracy that the DD FSM solution provides for tightly structured and executed strategies. This knowledge has come to light as a direct result of the research carried out in this Thesis and was not previously known.

In general, the application of Game AI techniques to fighting games as a means of mimicking human player strategies was an uncharted area in terms of research. This Thesis has identified and defined this novel problem domain and proposed multiple novel solutions that have been designed, implemented and evaluated. Particularly in the case of the DD FSM solution, the novel design presents a new way in which existing AI components can work together by passing data between levels to produce meaningful results when mimicking strategies. Previously, neither of the solutions had been implemented and tested to mimic strategies in a fighting game. However, now both solutions have been designed, implemented and have been proved to be successful via an evaluation.

9.4 Limitations and Future Work

While the results suggest that there is potential to incorporate such AI, as that described in Chapters 6 and 7, into fighting games to enable players to fight against mimicked versions of their rivals, what is not clear from this research is the impact this would have on the performance of commercial strategic fighting games. One weakness of this research is the fact that it has all been localised to a proof of concept game that has been tailored to encourage strategic play. There is a distinct lack of integration with a commercial fighting game that would further strengthen the argument that the techniques presented are fit for purpose. However, the proof of concept game does conform to rules found in typical commercial fighting games, while allowing for a more strategic approach towards gameplay.

The evaluation of the DD FSM and K nearest neighbour solutions, from within a commercial fighting game, was deliberately omitted from the scope of this project due to difficulties in obtaining code for commercial videogames. However, it should be noted that there is potential to conduct further research in this area. Having established from this project that the techniques work in terms of functionality within a proof of concept game, the next step would be to integrate the two techniques into a commercial strategic fighting game and gather performance metrics. The code would need to be tailored to conform to the rules of the commercial fighting game in question; however, the architecture must remain intact. Further functional testing within a commercial fighting game, that has different rules to the proof of concept game, or even another videogame genre entirely, would move the research closer to seeing a commercial solution come to fruition. For example, the solutions could be applied to sports games where players thought process and actions can be split into operations, tactics and strategies. Football videogames would provide a suitable test bed for this further research as players can pass, dribble, shoot and tackle at the operational level, combining these to develop tactics and ultimately strategies.

Limitations pertaining to the design and implementation of the techniques become apparent when reviewing the results of the experiment in Chapter 8. The DD FSM solution struggles to reproduce strategies where states change back and forth frequently under varying conditions, as was the case with strategy 5 in Table 8.4. The only way to avoid such anomalies in the reproduction of the strategy using the DD FSM solution is to ensure the source data collated during the human vs. human bouts is absolutely consistent across each play-through. However this is not always likely to be the case and gives rise to the underlying limitation: the DD FSM solution is highly prone to errors made by human players when they are attempting to reproduce their own strategy during the data capture phase. Future work to resolve this limitation would involve enhancing the algorithm described in Chapter 7 to remove noise and smooth the data to a higher standard.

This is a limitation that has a lesser impact on the k nearest neighbour solution in that this solution does not conform to any pre-defined structures, such as a DD FSM. However, this too leads to a limitation as the results documented in Chapter 8 suggest that the AI fighter often executed moves prematurely. This typically happens if one or more in-game parameter depletes quicker than it did during the human vs. human bouts. The nearest neighbour chosen by the KNN solution may have an output that was executed at a more advanced stage during the human vs. human bout, resulting in moves being prematurely executed by the AI fighter. The root cause of this limitation lies in the human fighter not conforming to their own strategy during the human vs. AI bout.

This thesis has been primarily focused on mimicking human strategies in fighting games, the next step would be to predict what the human player would do under circumstances that may not have been encountered. Both solutions presented in this thesis work provided both human players execute their strategies exactly during the data capture phase, and provided the human player follows their own strategy during the human vs. AI bout. However, if the human player during the human vs. AI bout breaks their own strategy by executing moves they would not normally carry out in a given situation, the AI fighter shall respond only in accordance to the data that has been captured already. The next step of this research would be to capture such responses and compare them against what the human would do if they were in the same situation. A player model could be built based on this and fed back into the AI solution, making for a more durable and adaptable opponent.

9.5 Thesis Summary and Final Conclusion

Two separate novel AI architectures for mimicking human player strategies in fighting games have been designed, developed and implemented using a proof of concept game. These architectures have been evaluated and have been proven to be successful across various sets of data. It is concluded that both the DD FSM solution and k nearest neighbour solution are equally capable of mimicking human strategies in fighting games, albeit across different strategies. The DD FSM solution performed well on basic strategies, but could not mimic strategies with numerous state transitions in quick succession. Further work can be carried out to resolve this by enhancing the quality of smoothing. This limitation was of less impact on the k nearest neighbour solution, which generally performed well, but prematurely executed moves if the human player during the human vs. AI bout broke their own strategy.

In terms of further research that can be carried out in the future by building on the contributions of this thesis, the focus now needs to shift to predicting outputs for unknown scenarios. Player modelling can be used to predict how a human player would potentially respond to an as yet un-encountered scenario. This could then be fed back to the AI solution to enhance performance and user experience, enabling humans partaking in the human vs. AI bout to learn more about their opponents and how to beat them.

References

ANAGNOSTOU, K., & MARAGOUDAKIS, M., 2009. Data mining for player modeling in videogames. In: *13th Panhellenic Conference on Informatics, 2009. Corfu, Sept 2009.* Los Alamitos: IEEE Computer Society, pp. 30-34.

ATKESON, C. G. & SCHAAL, S., 1997. Robot Learning from Demonstration. In: *Proceedings of IEEE International Conference on Robotics and Automation (ICRA97).* IEEE, pp. 1706-1712.

Badges of Fury, 2013. [Movie]. Directed by Wong TSZ-MING. Beijing Enlight Pictures, China and Hong Kong: Distributed worldwide by Easternlight Films.

BAR-COHEN, J., 2012. Nature as a Model for Mimicking and Inspiration of New Technologies. *International Journal of Aeronautical and Space Sciences*, 13(1), 1-13.

BARTO, A. G., & DIETTERICH, T. G., 2004. Reinforcement learning and its relationship to supervised learning. In: SI, J., BARTO, A. G., POWELL, W., & WUNSCH, D., eds. *Handbook of learning and approximate dynamic programming.* Wiley-IEEE Press, pp. 47-65.

BARTON, D., 2014. Using Case-Based Reasoning to Improve Real-Time Strategy Game AI. Unpublished.

BATALOV, D. V., & OOMMEN, B. J., 2001. On playing games without knowing the rules. In: *IFSA World Congress and 20th NAFIPS International Conference, 2001, Vancouver, July 2001.* pp. 1862-1868.

Battle Arena Toshinden, 1995. [PlayStation game]. TAMSOFT. Japan: distributed by Takara.

BELEW, R. K., MCINERNEY, J., & SCHRAUDOLPH, N. N., 1991. Evolving networks: Using the genetic algorithm with connectionist learning. In: LANGTON, C. G., TAYLOR, C., FARMER, J. D., RASMUSSEN. S.E., eds. *Artificial Life II: Proceedings of the Workshop on Artificial Life*. Reading: Addison Wesley, pp.511 -547.

BlazBlue: Chronophantasma, 2011. [PlayStation 3 game]. ARC SYSTEM WORKS. Japan: distributed by Aksys Games.

Blender, 2013. [PC software]. BLENDER FOUNDATION. Available from: http://www.blender.org/.

Bravely Default, 2013. [Nintendo 3DS game]. SQUARE ENIX. Japan: distributed by Nintendo.

BURO, M., & FURTAK, T., 2004. RTS games and real-time AI research. In: *Proceedings of the Behavior Representation in Modeling and Simulation Conference (BRIMS), Arlington, Virginia*. Curran Associates, pp.34-41.

CANT, R., CHURCHILL, J., & AL-DABASS, D., 2002. A hybrid Artificial Intelligence approach with application to games. In: *IJCNN'02. Proceedings of the 2002 International Joint Conference on Neural Networks, Honolulu, May 2002*. pp. 1575-1580.

CARNEIRO, E. M., MARQUES DA CUNHA, A., DIAS, L. A. V., 2014. Adaptive Game AI Architecture with Player Modeling. In: *2014 11th International Conference on Information Technology: New Generations*. IEEE, pp. 40-45.

CHAPEROT, B., & FYFE, C., 2005. Motocross and artificial neural networks. In: *Game Design and Technology Workshop 2005*.

CHAPEROT, B., & FYFE, C., 2006. Improving artificial intelligence in a motocross game. In: *2006 IEEE Symposium on Computational Intelligence and Games, Reno, May 2006.* IEEE Computational Intelligence Society, pp. 181-186.

CHARLES, D., MCNEILL, M., MCALISTER, M., BLACK, M., MOORE, A., STRINGER, K., KUCKLICH, J., KERR, A., 2005. Player-Centered Game Design: Player Modelling and Adaptive Digital Games. In: *Proceedings of the Digital Games Research Conference, Vancouver, June 2005.*

Chicken – an Egg exporter for Blender, 2006. [PC software]. SOURCEFORGE. Available from: http://sourceforge.net/projects/chicken-export/.

CHO, B. H., JUNG, S. H., SEONG, Y. R., & OH, H. R., 2006. Exploiting intelligence in fighting action games using neural networks. *IEICE Transactions on Information and Systems*, E89(3), 1249-1256.

CHO, B., PARK, C., & YANG, K., 2007. Comparison of AI Techniques for Fighting Action Games-Genetic Algorithms/Neural Networks/Evolutionary Neural Networks. In: *Entertainment Computing–ICEC 2007, 6th International Conference, Shanghai, September 2007*. *S*pringer, pp. 55-65.

COPPIN, B., 2004. *Artificial intelligence illuminated*. Jones & Bartlett Learning.

DANZI, G., SANTANA, A. H., FURTADO, A. W., GOUVEIA, A. R., LEITAO, A., & RAMALHO, G. L., 2003. Online adaptation of computer games agents: A reinforcement learning approach. In : *II Workshop de Jogos e Entretenimento Digital,* pp. 105-112.

DAVID, O. E., JAAP VAN DEN HERIK, H., KOPPEL, M. & NETANYAHU, N. S., 2014. Genetic Algorithms for Evolving Computer Chess Programs. *IEEE Transactions on Evolutionary Computation,* 18(5), 779-789.

Dead or Alive, 1997. [PlayStation game]. TEAM NINJA. Japan: distributed by TECMO.

Dead or Alive 2, 1999. [Arcade game]. TEAM NINJA. Japan: distributed by TECMO.

DI PIETRO, A., BARONE, L., WHILE, L., 2006. A Comparison of Different Adaptive Learning Techniques for Opponent Modelling in the Game of Guess It. In: *2006 IEEE Symposium on Computational Intelligence and Games, Reno, May 2006.* IEEE Computational Intelligence Society, pp. 173-180.

Diablo 3: Reaper of Souls, 2014. [PlayStation 3]. BLIZZARD ENTERTAINMENT. USA: distributed by Blizzard Entertainment.

DRACHEN, A., CANOSSA, A., & YANNAKAKIS, G. N., 2009. Player modeling using self-organization in Tomb Raider: Underworld. In: *2009 IEEE Symposium on Computational Intelligence and Games, Milano, September 2009.* IEEE Computational Intelligence Society, pp. 1-8.

Elder Scrolls Online, 2014. [PC game]. ZENIMAX ONLINE STUDIOS. USA: distributed by Bethesda Softworks.

Enter the Dragon, 1973. [Movie]. Directed by Robert CLOUSE. Golden Harvest and Concord Production Inc., USA and Hong Kong: distributed in UK by Warner.

ESPOSITO, N., 2005. A Short and Simple Definition of What a Video Game Is. In: *Proceedings of the Digital Games Research Conference*, *Vancouver, June 2005.*

FAIRCLOUGH, C., FAGAN, M., MAC NAMEE, B., & CUNNINGHAM, P., 2001. Research directions for AI in computer games. In: *Proceedings of the Twelfth Irish Conference on Artificial Intelligence & Cognitive Science, Maynooth, September 2001.* Maynooth: National University of Ireland, pp. 333–344.

FERNÁNDEZ, A., & GÓMEZ, S., 2008. Solving non-uniqueness in agglomerative hierarchical clustering using multidendrograms. *Journal of classification*, *25*(1), 43-65.

FIFA International Soccer, 1993. [Sega Genesis game]. EXTENDED PLAY PRODUCTIONS. Canada: distributed by Electronic Arts.

FIGLIOLINI, G. & CECCARELLI, M., 2002. A novel articulated mechanism mimicking the motion of index fingers. *Robotica,* 20(1), 13-22.

Final Fantasy VII, 1997. [PlayStation game]. SQUARESOFT. Japan: distributed by Sony Computer Entertainment Europe.

FINK, A., DENZINGER, J., & AYCOCK, J., 2007. Extracting NPC behavior from computer games using computer vision and machine learning techniques. In: *IEEE Symposium on Computational Intelligence and Games, Honolulu, April 2007*. pp. 24-31.

Fist of Fury, 1972. [Movie]. Direct by Lo WEI. Golden Harvest, Hong Kong: Distributed in UK by Hong Kong Legends.

Fist of Legend, 1994. [Movie]. Directed by Gordon CHAN. Eastern Productions, Hong Kong: Distributed worldwide by Golden Harvest.

GALLAGHER, M., & LEDWICH, M., 2007, April. Evolving pac-man players: Can we learn from raw input?. In: *IEEE Symposium on Computational Intelligence and Games, Honolulu,, April 2007.* pp. 282-287.

Game of Death, 1978. [Movie]. Directed by Bruce LEE. Golden Harvest and Concord Production Inc., Hong Kong: distributed in UK by Hong Kong Legends.

GINGOLD, Y. I., 2006. From Rock, Paper, Scissors to Street Fighter II: Proof by Construction. In: *Proceedings of the 2006 ACM SIGGRAPH symposium on Videogames, Boston, July 2006.* New York: ACM New York, pp. 155-158.

GRAEPEL, T., HERBRICH, R., & GOLD, J., 2004. Learning to fight. In: *Proceedings of the International Conference on Computer Games: Artificial Intelligence, Design and Education*. pp. 193-200.

HE, S., DU, J., CHEN, H., MENG, J., & ZHU, Q., 2008. Strategy-Based Player Modeling During Interactive Entertainment Sessions by Using Bayesian Classification. In: *ICNC'08. Fourth International Conference on Natural Computation, Jinan, October 2008.* Los Alamitos: IEEE Computer Society, pp. 255-261.

HE, S., WU, G., MENG, J., CHEN, H., LI, J., LIU, Z., & ZHU, Q., 2008. Game Player Strategy Pattern Recognition by Using Radial Basis Function. In: *International Conference on Computer Science and Software Engineering, Wuhan, December 2008.* Los Alamitos: IEEE Computer Society, pp. 937-940.

HE, S., WANG, Y., XIE, F., MENG, J., CHEN, H., LUO, S., LIU, Z., & ZHU, Q., 2008. Game Player Strategy Pattern Recognition and How UCT Algorithms Apply Pre-Knowledge of Player's Strategy to Improve Opponent AI. In: *2008 International Conference on Computational Intelligence for Modelling Control & Automation, Vienna, December 2008.* Los Alamitos: IEEE Computer Society, pp. 1177-1181.

Heavyweight Champ, 1976. [Arcade game]. SEGA. Japan: distributed by Sega.

HILDMANN, H., & CROWE, M., 2011. Behavioural game AI - A theoretical approach. In: *2011 International Conference for Internet Technology and Secured Transactions (ICITST), Abu Dhabi, December 2011.* pp.550-555.

HISASHI, H., 2014. Deep Boltzmann Machine for Evolutionary Agents of Mario AI. In: *2014 IEEE Congress on Evolutionary Computation (CEC)*, *Beijing, China, July 2014.* IEEE, pp. 36-41.

HOLLAND, J., 1984. Genetic algorithms and adaptation. In: SELFRIDGE, O.G., RISSLAND, E.L., ARBIB, M.A., eds*. Adaptive Control of Ill-Defined Systems.* New York: Plenum Press, pp317-333.

HONG, J. H., & CHO, S. B., 2004. Evolution of emergent behaviors for shooting game characters in Robocode. In: *Congress on Evolutionary Computation, 2004.* pp. 634-638.

HOULETTE, R., 2003. Player Modelling for Adaptive Games. In: RABIN, S., ed. *A.I Game Programming Wisdom 2.* Hingham, MA: Charles River Media, pp. 557-566.

HOULETTE, R., & FU, D., 2003. The Ultimate Guide to FSMs in Games. In: RABIN, S., ed. *A.I Game Programming Wisdom 2.* Hingham, MA: Charles River Media, pp. 283-301.

HSIEH, J. L., & SUN, C. T., 2008. Building a player strategy model by analyzing replays of real-time strategy games. In: *IEEE International Joint Conference on Neural Networks, 2008, IJCNN 2008. (IEEE World Congress on Computational Intelligence), Hong Kong, June 2008.* pp. 3106-3111.

HYDE, D., 2008. Using Bayesian Networks to Reason About Uncertainty. In: RABIN, S., ed. *A.I Game Programming Wisdom 4*. Boston: Cengage Learning, pp. 429-441.

Ip Man, 2008. [Movie]. Directed by Wilson YIP. Mandarin Films, Hong Kong: Distributed worldwide by Mandarin Films.

Ip Man 2, 2010. [Movie]. Directed by Wilson YIP. Mandarin Films, Hong Kong: Distributed worldwide by Mandarin Films.

JOHNSON, D., & WILES, J., 2001. Computer games with intelligence. In: *The 10th IEEE International Conference on Fuzzy Systems, Melbourne, Dec 2001.* pp. 1355-1358.

Karate Champ, 1984. [Arcade game]. TECHNOS JAPAN. Japan: distributed by Data East.

KAUKORANTA, T., SMED, J., & HAKONEN, H., 2003. Understanding Pattern Recognition Methods. In: RABIN, S., ed. *A.I Game Programming Wisdom 2*. Hingham, MA: Charles River Media, pp. 579-589.

KAUKORANTA, T., SMED, J., & HAKONEN, H., 2003. Role of pattern recognition in computer games. In: *Proceedings of the 2nd International Conference on Application and Development of Computer Games, Hong Kong, January 2003.* Hong Kong: City University, pp. 189-94.

KAWAMURA, Y., 2010. Experimental Studies on Various Types of Animal Mimicking Robots. In: IEEE *2010 International Conference on Broadband, Wireless Computing, Communication and Applications, November 2010.* IEEE, pp. 755-759.

KELLER, F., n.d. Clustering. *Computer University Saarlandes Tutorial Slides*

KENYON, S. H., 2006. Behavioral software agents for real-time games. *Potentials, IEEE*, 25(4), 19-25.

KHOO, A., & ZUBEK, R., 2002. Applying inexpensive AI techniques to computer games. *Intelligent Systems, IEEE*, *17*(4), 48-53.

LAIRD, J. E., 2001. Using a computer game to develop advanced AI. *Computer*, *34*(7), 70-75.

LEE, B., 1975. *Tao of Jeet Kune Do.* Burbank, CA: Ohara Publications.

LEE, G., BULITKO, V. & LUDVIG, E. A., 2014. Automated Story Selection for Color Commentary in Sports. *IEEE Transactions on Computational Intelligence and AI in Games*, 6(2), 144-155.

LEE, L., 2005. *Adaptive Behavior for Fighting Game Characters*. Unpublished thesis (MSc.), San Jose State University.

LENT, M., 2007. Game Smarts. *Computer-IEEE Computer Magazine,* 40(4), 99-101.

LIU, S., LOUIS, S. J. & BALLINGER, C., 2014. Evolving Effective Micro Behaviors in RTS Game. In: *2014 IEEE Conference on Computational Intelligence and Games (CIG).* IEEE, pp. 1-8.

LU, F., YAMAMOTO, K., NOMURA, L. H., MIZUNO, H., LEE, Y. & THAWONMAS, R., 2013. Fighting Game Artificial Intelligence Competition Platform. In: *2013 IEEE 2nd Global Conference on Consumer Electronics, October 2013.* IEEE, pp. 320-323.

LUCAS, S. M., & KENDALL, G., 2006. Evolutionary computation and games. *Computational Intelligence Magazine, IEEE*, *1*(1), 10-18.

LUEANGRUEANGROJ, S., KOTRAJARAS, V., 2009. Real-time Imitation based Learning for Commercial Fighting Games. In: *Proceedings of International Conference on Computer Games, Multimedia and Allied Technology 2009.*

MALLOUK, W., & CLUA, E., 2006. An Object-Oriented Approach for Hierarchical State Machines. In: *Proceedings of the SBGames conference in Computing, Recife, November 2006.* pp. 8-10.

Man of Tai Chi, 2013. [Movie]. Directed by Keanu REEVES. Universal Studios, China and USA: Distributed by Universal Pictures.

Marvel vs. Capcom 2, 2000. [Sega Dreamcast game]. CAPCOM. Japan: distributed by Capcom.

Marvel vs. Capcom 3, 2011. [PlayStation 3 game]. CAPCOM. Japan: distributed by Capcom.

McCARTHY, J., 2002. *What is artificial intelligence?* [online]. [viewed 19/01/2013]. Available from: http://bcrc.bio.umass.edu/courses/fall2008/biol/biol270h/3-Discussions/13-ET_Intelligence/13f-AI/13f-1_AI-Intro.pdf.

MILES, C., QUIROZ, J., LEIGH, R., & LOUIS, S. J., 2007. Co-evolving influence map tree based strategy game players. In: *IEEE Symposium on Computational Intelligence and Games, 2007. CIG 2007, Honolulu, April 2007.* pp. 88-95.

Mortal Kombat, 1992. [Super Nintendo Entertainment System game]. MIDWAY. USA: distributed in UK by Virgin.

Mortal Kombat, 2011. [PlayStation 3 game]. NETHERREALM STUDIOS. USA: distributed by Warner Bros.

Mortal Kombat 3, 1995. [Super Nintendo Entertainment System game]. MIDWAY. USA: distributed Williams Entertainment.

Mortal Kombat Trilogy, 1996. [PlayStation game]. MIDWAY. USA: distributed Williams Entertainment.

MotioninJoy, 2012. [PC software]. MOTIONINJOY.COM. Available from: http://www.motioninjoy.com/.

MOUCHET, A., 2005. Subjectivity in the articulation beetween strategy and tactics in team sports: an example in rugby. *Italalian Journal of Sport Sciences,* 12(1), 24-33.

MultiDendrograms, 2008. [PC software]. FERNÁNDEZ, A., & GÓMEZ, S. Available from: http://deim.urv.cat/~sergio.gomez/multidendrograms.php.

MUSASHI, M. & HARRIS, V., 1974. *A Book of Five Rings*. New York: Overlook.

n.d. *Untitled Document.* [online]. University of Maribor. [viewed 06 Jan 2014]. Available from: http://www.ro.feri.uni-mb.si/predmeti/int_reg/Predavanja/Eng/2.Neural%20networks/_06.html

NAREYEK, A., 2007. Game AI is dead. Long live game AI!. *Intelligent Systems, IEEE*, *22*(1), 9-11.

ORTIZ B, S., MORIYAMA, K., FUKUI, K. I., KURIHARA, S., & NUMAO, M., 2010. Three-subagent adapting architecture for fighting videogames. In: ZHANG, B.T., ORGUN, M.A., eds. *PRICAI 2010: Trends in Artificial Intelligence*. Berlin: Springer, pp. 649-654.

Pac-Man, 1980. [Arcade game]. NAMCO. Japan: distributed by Midway.

Panda3D, 2010. [PC software]. CARNEGIE MELON UNIVERSITY.

PEDERSEN, C., TOGELIUS, J., & YANNAKAKIS, G. N., 2009. Modeling player experience in Super Mario Bros. In: *IEEE Symposium on Computational Intelligence and Games, 2009. CIG 2009, Milano, September 2009.* IEEE Computational Intelligence Society, pp. 132-139.

Persona 4 Arena Ultimax, 2014. [PlayStation 3 game]. ARC SYSTEM WORKS. Japan: distributed in UK by Sega.

PyGame, 2008. [PC software]. PYGAME.ORG. Available from: http://www.pygame.org/download.shtml.

PyPad 360, 2010. [PC software]. RABID SQUIRREL GAMES. Available from: https://code.google.com/p/mazeofnight/source/browse/trunk/Prototype/pyPad360.py?r=51.

Quake, 1996. [PC game]. ID SOFTWARE. USA: distributed by GT Interactive.

RAMIREZ, A. & BULITKO, V., 2014. Automated Planning and Player Modelling for Interactive Storytelling. *IEEE Transactions on Computational Intelligence and Artificial Intelligence in Games*, *Unpublished.*

RICCIARDI, A., & THILL, P, 2008. *Adaptive AI for Fighting Games*. [online]. [viewed 13/01/2013]. Available from: http://cs229.stanford.edu/proj2008/RicciardiThill-AdaptiveAIForFightingGames.pdf.

RICH, E., KNIGHT, K., 1991. *Artificial Intelligence*. New York: McGraw-Hill.

RIEK, L. D. & ROBERTSON, P., 2008. Real-Time Empathy: Facial Mimicry on a Robot. In: *Workshop on Affective Interaction in Natural Environments (AFFINE) at the International ACM Conference on Multimodal Interfaces (ICMI 08).* ACM.

ROBERTSON, G. & WATSON, I., 2014. A Review of Real-Time Strategy Game AI. Unpublished.

ROSADO, G., 2003. Implementing a Data-Driven Finite State Machine. In: RABIN, S., ed. *A.I Game Programming Wisdom 2.* Hingham, MA: Charles River Media, pp. 307-317.

SAINI, S., CHUNG, P. W. H., & DAWSON, C. W., 2011. Mimicking human strategies in fighting games using a Data Driven Finite State Machine. In: *2011 6th IEEE Joint International Information Technology and Artificial Intelligence Conference (ITAIC), Chongqing, August 2011.* Beijing: IEEE, pp. 389-393.

SAINI, S. S., DAWSON, C. W., & CHUNG, P. W., 2011. Mimicking player strategies in fighting games. In: *2011 IEEE International Games Innovation Conference (IGIC), Orange County, November 2011.* IEEE, pp. 44-47.

SANTOSO, S. & SUPRIANA, I., 2014. Minimax Guided Reinforcement Learning for Turn-based Strategy Games. In: *2014 2nd International Conference on Information and Communication Technology (ICoICT), May 2014.* IEEE, pp. 217-220.

SCHAAL, S., 1997. Learning From Demonstration. In: MOZER, M. C., JORDAN, M., PETSCHE, T., eds. *Advances in Neural Information Processing Systems 9.* Cambridge, MA: MIT Press.

Street Fighter, 1987. [Arcade game]. CAPCOM. Japan: distributed by Capcom.

Street Fighter 4, 2008. [Arcade game]. CAPCOM. Japan: distributed by Capcom.

Street Fighter II, 1991. [Super Nintendo Entertainment System game]. CAPCOM. Japan: distributed by Capcom.

Street Fighter Alpha, 1995. [PlayStation game]. CAPCOM. Japan: distributed by Capcom.

Street Fighter Alpha 3, 1998. [PlayStation game]. CAPCOM. Japan: distributed by Capcom.

Street Fighter X Tekken, 2012. [PlayStation 3 game]. CAPCOM. Japan: distributed by Capcom.

Street Fighter Zero 3, 1998. [PlayStation game]. CAPCOM. Japan: distributed by Capcom.

Super Mario Bros, 1985. [Nintendo Entertainment System game]. NINTENDO. Japan: distributed by Nintendo.

Super Street Fighter II Turbo, 1994. [PC game]. CAPCOM. Japan: distributed by Capcom.

Tales of Xillia 2, 2014. [PlayStation 3 game]. BANDAI NAMCO GAMES. Japan: distributed by Bandai Namco Games.

Tekken 3, 1997. [Arcade game]. NAMCO. Japan: distributed by Namco.

Tekken 4, 2002. [PlayStation 2 game]. NAMCO. Japan: distributed by Namco.

Tekken Tag Tournament, 2000. [PlayStation 2 game]. NAMCO. Japan: distributed by Sony Computer Entertainment Europe.

Tekken Tag Tournament 2, 2012. [PlayStation 3 game]. BANDAI NAMCO GAMES. Japan: distributed by Bandai Namco Games.

The Expendables, 2010. [Movie]. Directed by Sylvester STALLONE. Millennium Films and Nu Image, USA: Distributed worldwide by Lionsgate.

The Protector 2, 2014. [Movie]. Directed by Prachya PINKAEW. Sahamongkolfilm International, Thailand: Distributed by Sahamongkolfilm International.

The Raid, 2011. [Movie]. Directed by Gareth EVANS. PT. Merantau Films in association with MNC Pictures and XYZ Films, Indonesia: Distributed worldwide by Celluloid Nightmares.

The Raid 2, 2014. [Movie]. Directed by Gareth EVANS. PT. Merantau Films and XYZ Films, Indonesia: Distributed worldwide by Celluloid Nightmares.

Thief, 1998. [PC game]. LOOKING GLASS STUDIOS. USA: distributed by Eidos Interactive.

THUNPUTTARAKUL, W. & KOTRAJARAS, V., 2007. Data Analysis for Ghost AI Creation in Commercial Fighting Games. In: *Proceedings of GAMEON 2007, Bologna, November 2007.* Eurosis, pp. 37-41.

Tomb Raider Underworld, 2008. [PlayStation 3 game]. CRYSTAL DYNAMICS. USA: distributed by Eidos Interactive.

TURNER, K., & VAN SCHUYVER, M., 1991. *Secrets of Championship Karate*. Chicago: Contemporary.

TZU, S., 1910. *The Art of War*. Translated from Chinese by Lionel Giles.

Ultra Street Fighter IV, 2014. [PlayStation 3 game]. CAPCOM. Japan: distributed by Capcom.

VAN SCHUYVER, M. & VILLALOBOS, P. S., 2002. *Fighting Strategies of Muay Thai: Secrets of Thailand's Boxing Camps*. Boulder, CO: Paladin.

Warcraft, 1994. [PC game]. BLIZZARD ENTERTAINMENT. USA: distributed by Blizzard Entertainment.

XIULI, Z., HAOJUN, Z., PENG, L. & GUANGMING, L., 2006. Designing a Quadrupedal Robot Mimicking Cat Locomotion. In: *2006 IEEE International Conference on Systems, Man, and Cybernetics, Taipei, Taiwan, October 2006.* IEEE, pp. 4471-4474.

X-Men vs. Street Fighter, 1996. [Arcade game]. CAPCOM. Japan: distributed by Capcom.

YAMAMOTO, K., MIZUNO, S., CHU, Y. C., THAWONMAS, R., 2014. Deduction of Fighting-Game Countermeasures Using the k-Nearest Neighbor Algorithm and a Game Simulator. In: *2014 IEEE Conference on Computational Intelligence and Games (CIG), August 2014.* IEEE, pp. 1-5.

YAU, Y. J., & TEO, J., 2006. An empirical comparison of non-adaptive, adaptive and self-adaptive co-evolution for evolving artificial neural network game players. In: *2006 IEEE Conference on Cybernetics and Intelligent Systems, Bangkok, June 2006.* IEEE, pp. 1-6.

Yie Ar Kung-Fu, 1985. [Arcade game]. KONAMI. Japan: distributed by Konami.

ZHOU, C. N., YU, X. L., SUN, J. Y., & YAN, X. L., 2006. Affective Computation Based NPC Behaviors Modeling. In: *Proceedings of the 2006 IEEE/WIC/ACM international conference on Web Intelligence and Intelligent Agent Technology Workshops, 2006. WI-IAT 2006 Workshops, Hong Kong, December 2006.* Los Alamitos: IEEE Computer Society, pp. 343-346.

Appendix A.1 – Move List

Table A.1 below shows the character representation of moves that can be made in the proof of concept game. This representation is used when spooling data from the bouts and is the same notation used in tables throughout this thesis.

Table A.1 – Move Representation

Move	Key
Jab	j
Cross	c
Right Hook	r
Left Hook	l
Uppercut	u
Haymaker	h
Right Body Shot	n
Left Body Shot	m
Short Jab	1
Short Cross	2
Back Evasion	a
Left Evasion	q
Right Evasion	w
Push	3
Block	b
Low Block	p
Low Kick	5
Sidekick	6
Forward Lunge	x
Backwards Lunge	z

Appendix A.2 – Sample Human Transcript

The following table contains sample data from the human vs. human bout of strategy 1 as articulated in Table 8.3. Such transcripts exist for all strategies in Table 8.3.

Table A.2 – Sample Human Transcript

Opponent Player			Player to be mimicked			Distance	Moves	Play through
Health	Morale	Stamina	Health	Morale	Stamina			
100	50	100	100	50	99	5.210001	j	1
100	50	100	100	50	98	5.210001	j	1
100	50	100	100	50	97	4.310001	j	1
98	50	98	98	50	95	4.850002	j	1
97	50	97	98	50	95	4.850002	*j*	1
97	50	97	97	50	94	4.850002	j	1
97	50	97	97	50	93	4.760002	j	1
96	50	97	97	50	92	4.130002	j	1
95	50	96	97	50	92	4.130002	*j*	1
95	50	96	96	50	91	4.130002	j	1
94	50	95	96	50	91	4.130002	*j*	1
94	50	95	95	50	90	4.130002	j	1
86	50	85	89	50	79	3.860002	*3*	1
79	50	54	68	50	67	4.820003	b	1
79	50	54	68	50	67	4.820003	b	1
79	50	54	68	50	67	4.820003	b	1
79	50	54	68	50	67	4.820003	b	1
79	50	54	68	50	67	4.820003	b	1
79	50	10	68	50	50	4.460002	b	1
79	50	10	68	50	50	4.460002	b	1
79	50	10	68	50	50	4.460002	q	1
79	50	10	68	50	50	4.460002	b	1
79	50	4	66	50	47	3.470004	u	1
75	50	2	66	50	47	3.470004	*c*	1
75	50	2	66	50	45	3.470004	u	1
71	50	0	66	50	45	3.470004	*c*	1

Opponent Player			Player to be mimicked			Distance	Moves	Play through
Health	Morale	Stamina	Health	Morale	Stamina			
71	50	0	66	50	43	3.470004	u	1
67	50	-1	66	50	43	3.470004	*j*	1
67	50	-1	66	50	42	3.470004	5	1
63	50	-4	66	50	41	3.470004	*j*	1
63	50	-4	66	50	39	3.470004	u	1
59	50	-4	66	50	37	3.470004	u	1
55	50	-5	66	50	37	3.470004	*j*	1

Appendix A.3 – Sample DD FSM Transcript

The following table contains sample data from the human vs. DD FSM AI bout of strategy 1 as articulated in Table 8.3. Such transcripts exist for all strategies in Table 8.3.

Table A.3 – Sample DD FSM Transcript

Opponent Player			AI Player			Distance	State	Moves
Health	Morale	Stamina	Health	Morale	Stamina			
100	50	100	100	50	100	5.660001	s2	_ u_
100	50	98	96	50	100	4.490001	s2	_ j j_
100	50	96	94	50	100	4.490001	s2	_ c j_
100	50	93	91	50	100	5.300004	s2	_ j c_
99	50	90	88	50	98	4.850004	s2	_ j_
98	50	89	87	50	97	4.850004	s2	_ j c_
82	50	79	77	50	82	4.670002	s2	_ j j_
80	50	77	75	50	79	4.670002	s2	_ j j_
65	50	63	59	50	63	3.590002	s2	_ j j_
64	50	61	57	50	62	4.490002	s2	_ u_
64	50	59	53	50	62	3.500002	s2	_ j j_
64	50	57	51	50	62	4.490002	s2	_ j_
59	50	56	50	50	58	4.490002	s1	_ u_
59	50	54	46	50	57	4.040002	s1	_ b b b b b b_
58	50	52	46	50	54	4.670002	s1	_ b b b b b b_
58	50	48	46	50	51	4.670002	s1	_ b b b b_
58	50	44	46	50	47	4.670002	s2	_ b b b_
58	50	41	46	50	45	4.490002	s2	_ 1 1_
57	50	39	42	50	42	3.680002	s2	_ 1_
53	50	38	40	50	37	3.410002	s2	_ 1 1_
47	50	36	36	50	34	3.410002	s2	_ 1_
47	50	35	34	50	29	4.220003	s2	_ 1_
33	50	34	32	50	20	4.760001	s2	_ 1 j u_
32	50	30	31	50	19	4.940001	s2	_ u_

Opponent Player			AI Player			Distance	State	Moves
Health	Morale	Stamina	Health	Morale	Stamina			
31	50	28	31	50	17	4.490002	s2	_ 1 j_
31	50	26	30	50	16	4.940002	s2	_ u_
31	50	24	30	50	16	4.940002	s2	_ u_
28	50	22	30	50	15	4.220002	s2	_ 1 u_
28	50	19	28	50	15	4.670002	s2	_ 1_
28	50	18	26	50	15	3.680002	s2	_ 1_
28	50	17	24	50	15	4.760002	s2	_ u 1_
24	50	2	20	50	1	3.590004	s2	_ 1 u_

Appendix A.4 – Sample KNN Transcript

The following table contains sample data from the human vs. K nearest neighbour AI bout of strategy 1 as articulated in Table 8.3. Such transcripts exist for all strategies in Table 8.3.

Table A.4 – Sample KNN Transcript

Opponent Player			AI Player			Distance	Moves
Health	Morale	Stamina	Health	Morale	Stamina		
100	50	100	100	50	100	8	__
100	50	100	100	50	100	5.570001	_ u_
100	50	98	96	50	100	4.580001	_ j j_
100	50	96	94	50	100	4.580001	_ j c_
100	50	93	91	50	100	4.580001	_ j_
100	50	92	90	50	100	4.580001	_ c j_
100	50	89	87	50	99	5.030005	_ j j_
97	50	87	85	50	93	4.490004	_ j_
74	50	86	84	50	77	5.210003	_ j_
73	50	85	83	50	75	5.660001	_ j j_
70	50	83	81	50	70	4.760001	_ j_
67	50	82	80	50	62	4.130001	_ c_
64	50	80	78	50	58	4.580002	_ b_
64	50	80	78	50	56	5.570002	_ c c_
62	50	76	74	50	51	4.490002	_ b b b b b b_
61	50	70	74	50	45	4.580002	_ j b_
60	50	67	73	50	44	4.580002	_ b b b b b b_
60	50	61	73	50	40	4.580002	_ b b b b b b b b_
60	50	57	73	50	36	4.580002	_ b b b b b b_
60	50	53	73	50	33	4.580002	_ u_
58	50	50	69	50	32	4.130001	_ u u_
58	50	46	61	50	30	3.590001	_ 1 1_
58	50	44	57	50	30	3.590001	_ 1 1_
50	50	42	53	50	24	3.410001	_ 1 1_
46	50	40	49	50	19	3.410001	_ n 1_
44	50	37	43	50	17	3.410001	_ 1_

Opponent Player			AI Player			Distance	Moves
Health	**Morale**	**Stamina**	**Health**	**Morale**	**Stamina**		
34	50	30	29	50	2	3.860001	_ 1_
30	50	29	27	50	-2	3.410002	_ m_
30	50	27	23	50	-2	3.680001	_ m m_
30	50	23	15	50	-2	3.680001	_ m_
30	50	21	11	50	-2	3.680001	_ m m_
30	50	17	3	50	-2	3.680001	_ u_
30	50	15	-1	50	-2	3.680001	_ u u_