General Agent Learning Using First Person Shooter Game Logs

by

Matthew P. Sheehan

The University of Auckland

Supervised by Dr. Ian Watson

A thesis submitted in fulfillment
of the requirements for the degree of

Master of Science

Department of Computer Science
The University of Auckland, New Zealand

July 2008

Abstract

Interactive computer game logs show the potential for use as replacement for time-consuming supervisory learning processes for embodied, situated agents. Unique, motivated, expert-level behavioural data is encoded within these logs, which if extracted could be of use to model a general agent. The research is distinctive in attempting to use 'ready-made' game logs downloaded directly from the internet, made by expert-level players, to take advantage of the convenience these logs provide for researchers: not having to prepare agent training examples. In the course of the research, it was found supervisory levels are needed to be kept low as it is easier to produce single logs and elucidate every action separately in (basic) movements for learning agents than it is to extract behaviours from the readily available, expert-level game logs by hand. There are issues in this process that need to be overcome, shown in an unsuccessful attempt to use largely rule-based data mining processes to learn these behaviours from the game logs. It was found the inherently top-down and statistical nature of such processes were fundamentally at odds with the unsupervised bottom-up and causal learning requirement of the problem. The innate issues with using game logs toward the goal of unsupervised agent learning are discussed. Possible approaches to the problem subsuming successful applications of various methods in narrower fields are presented for both symbolic and sub-symbolic advocates.

Acknowlegements

Firstly, I would like to thank my supervisor, Ian Watson, for his advice and help throughout the past year. Secondly, thanks to Patricia Riddle and Mike Barley for their willingness for discussion. Finally, I would like to thank my family and friends who have supported and encouraged me through this time.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This research primarily addresses the use of downloaded, expert-level, first-person shooter (FPS) interactive computer game logs for embodied, task-generalised, goal-oriented, automated-player (agent) learning. In particular, it investigates the potential of these 'off the shelf' game logs to supplement or even replace time-consuming supervisory learning methods. The original hypothesis of the research was to use rule-based data mining methods to investigate this potential, in finding and extracting rules that could point towards a method of actual agent implementation. This chapter will first provide a background to this research, starting with a brief outline of early game, board game and computer game artificial intelligence (AI) in section 1.1. Next, the main concepts and parameters of this research will be presented (section 1.2). Finally, an overview of the aims and objectives of this study will be outlined (section 1.3).

1.1 Games and AI

The potential for agent learning using computer games (i.e. video games) has been recognised for some time in the academic AI community, based in part on the legacy of early AI programs using board games such as checkers, chess and backgammon. Like those early programs, agents modelled using early computer games faced complex challenges that also tested their decision-making abilityb but in complicated, uncertain environments. In more recent, real-time games, such as those in the first-person shooter (FPS) genre, the challenges faced by agents necessitate a broad range of skills to survive in often frenetic, complex, three-dimensional environments. This section begins with an overview of classic (board) game AI (section 1.1.1). Following this is an introduction to AI in the computer games industry and its connection to the academic community (section 1.1.2).

1.1.1 Classic (Board) Game AI

Using games as a research tool for exploring new ideas in artificial intelligence is not new: Both Shannon and Turing in the 1940s and 1950s wrote chess playing programs (Shannon used a minimax procedure to decide new moves by use of a positional evaluation function; Turing was also involved in tic-tac-toe and checkers programs) and Samuels in the 1950s and 1960s used a checkers playing program for research. Samuels' checkers player included improving positional evaluation accuracy with machine learning techniques using some of the earliest non-numerical computational techniques. Research since (Schaeffer (2000) mentions work on the alpha-beta search algorithm with chess in particular) has seen enormous advances in the field, with quite public results: 1997 saw the toppling of human player world champions by computer programs in Othello (Takeshi Murakami defeated by Michael Buro's 'Logistello'), Checkers (Marion Tinsley resigned sick against Jonathan Schaeffer's 'Chinook'), and perhaps most famously Deep Blue's (Hsu, Campbell, Hoane et al (IBM)) wins over Gary Kasparov. While some classic games (most notably 'Go' (Arthur, 2006)) remain lagging in expert human play, even games of imperfect information like poker have seen successes against top human competition (Olsen, 2007).

1.1.2 Computer Game Industry AI

The evolution of the computer game industry's AI from its beginnings in the late 1970s to the mid 1990s did not improve much: the simple state-machine techniques used in Pac-man were re-used in various forms over this period to produce enemies that invariably charged directly at the player when seen (Millington, 2006). The AI in the game industry has improved greatly since then, with some games being marketed on the quality of their AI, including some games with integrated machine learning components (e.g. 'Black & White' (Lionhead Studios, 2001), 'The Sims' (Maxis, 2000), 'F.E.A.R' (Monolith Productions, 2005)).

However, AI programmers within the industry are still hamstrung by a number of issues, perhaps most notoriously being that the AI component of any game is the last component to be implemented, when deadlines and budgets have already been stretched (Millington, 2006). Another consideration that must be made is to the overarching necessity to keep the player within the narrative created by the game, which means the illusion of intelligence in the game's agents is often more important than

(functional intelligence. For instance, a predictable agent that is relatively simple would be preferred to an agent applying state-of-the-art techniques that could behave unpredictably and destroy this illusion. This can be equated in academic terms as relying on the 'Eliza Effect' (Weizenbaum, 1966), "...in which people see subtlety, understanding, and emotion in an agent as long as the agent does not actively destroy the illusion" (Bates, 1991). These issues at the developmental stage are facing increasing challenges from consumers with, as Laird and van Lent (2000) has noted especially of the FPS genre, "better AI becoming the point of comparison" between games. A brief outline of the FPS genre is outlined next, as it has particular relevance to this thesis.

FPS Games AI

The first-person shooter (FPS) game genre depends on real-time, first-person AI tactics (combat AI) more than any other genre: Schreiner (2002) went as far as to say that "First Person Shooters depend on combat AI to make their games playable, let alone interesting". However, even with such dependence, like the rest of the computer gaming industry, FPS game developers were slow to recognise the importance of AI. Schreiner (2002) in particular noted that "...the evolution of combat AI was slow for many years. Most AI opponents were relegated to shambling forward, often right into the player's gunfire".

While Millington (2006) thought that it was not until Rare Ltd's 'GoldenEye 007' (Rareware, 1997) (which had characters that could see and recognise when colleagues were killed) that the gaming public was made aware of what AI could do to improve their gaming experience, Schreiner (2002) believed it was the combat tactics of the marines in Valve Software's 'Half-Life' (Valve Software, 1998) that saw the turning point for combat AI in the FPS genre. It was also around this time the creators of a number of FPS games (including Half-Life) began to release portions or the entirety of their game code, in a bid to extend the lives of these games through user-generated content. The open access to the software, exciting first-person detailed environments, embedded fledgling AI components and increasing computing power sparked real interest in the academic AI community (Laird and van Lent, 2000).

Links to, and Motivations for Academic AI

While the industry's AI development had been constrained, the size and inherently competitive nature of the computer game industry has led to increasingly intricate

graphical interfaces, providing increasingly realistic gaming environments for the agents situated within. While there had been some examples of the use of computer games for AI research before, notably (Agre and Chapman, 1987; Fasciano, 1996; McCarthy, 1998), it was Laird and van Lent (2000) who noted that these environments, coupled with their open-source nature and fledgling AI components would provide the perfect test-bed for AI research. In return they believed that academic AI could help the computer game industry by testing and trying new techniques in these environments:

> "as development budgets soar, as companies get more risk averse, and as technology development costs need to be spread over more titles; having a reliable toolkit of tried-and-tested techniques [for industry developers] is the only sane choice" Millington (2006, pg. xxxii).

This period marks the point where research starts to be directly relevant to the current research. For further examples, with a particular bias towards the FPS genre, see the Literature Review in Chapter 3.

1.1.3 Data Mining and Data Mining in Games

Witten and Frank (2005, pg. 5) have used a definition of data mining of "the process of discovering patterns in data" noting "Useful patterns allow us to make nontrivial predictions on new data". The explosion of recorded information in the computer age has seen real growth in the field of data mining (Pyle, 1999). The use of data mining to search for behavioural patterns in data has been seen by Agrawal et al. (1993), who used data examples from supermarket shoppers to find rules between purchased items. One of the first examples of using data mining to find patterns in games was by Bhandari et al. (1997), who found rules for basketball teams from recorded data. From a computer games industry perspective, "the primary purpose of data mining in games is to find patterns of behavior, structure or content in order to improve the overall gameplay, hence keeping players longer and increasing the revenue of the game service" (Tveit and Tveit, 2002) rather than to use it to study artificial agents. For examples of the rule-based data mining methods used in this research, see Chapter 2. Examples of the use of data mining for behavioural pattern search in computer games are outlined in Chapter 3. The next section 1.2 below begins to outline some of the concepts that feature prominently in this thesis.

1.2 Outline of Concepts

This section presents some of the major concepts discussed in this thesis, doubling as both a means of introduction for perhaps unfamilar concepts and means of clarification for terms with several possible meanings.

1.2.1 Computer Game Concepts

Computer Game

The term 'computer game' in this thesis refers to the category of commercial video games played in virtual worlds. This is in contrast to computer simulations of board games such as chess and checkers.

Player

A 'player' in this thesis generally implies a person who plays computer games.

First-Person Shooter Games

First-person shooter (FPS) games are a sub-genre of action computer games in which the action is focussed on the use of (usually ranged) firearms from a first-person perspective. FPSs can be single or multi-player and can involve a large array of (often fantastical) weaponry and opponents. Examples of FPSs from this research include Team Fortress Classic, a multi-player version of the game Half-Life (Valve Software, 1998) and Quake II (id Software Inc., 1997).

Team Fortress Classic

Team Fortress Classic (TFC) is a multi-player FPS that pits two teams of players against each other in numerous capture-the-flag scenarios. Game points are allocated each time an opposing team's flag is returned to home base. When joining a game, players can choose between nine different classes of player, shown in Figure 1.1. Classes have different attributes and specific weaponry which make them useful in different situations. Because the game is as much about protecting the home team's flag as capturing the opposing teams, a mixture of fast, lightly weaponed classes for flag capture is often balanced by slower, heavier weaponed classes for defensive work and covering fire.

Figure 1.1: The nine different character classes in TFC: From top left: Demo-man, Engineer, Heavy Weapons Guy, Medic, Pyro. From bottom left: Scout, Sniper, Soldier and Spy

Armour, ammunition and health items ('powerups') are distributed around each different world environment ('map') the game can be played on, and can be 'picked up' by being walked over. If a player is shot, some of the player's health is lost. Once down to zero, the player is killed. A player can be healed by walking over a health pack or by a medic class character of his/her team. If a player collects an armour powerup, the armour will be damaged and destroyed before the player's health begins to be affected.

Quake II

QUAKE II can be played as both a single and multi-player game. The focus of this research is on the multi-player version, which like TFC can pit teams of players against each other, but unlike TFC there is an option for several individuals to all fight each

other. Points are awarded for each opponent kill and a point is lost every time a player kills himself or a team mate. Though different in-game character (avatar) appearances ('skins') can be used, there are no separate character classes.

As well as armour and health powerups, a player in QUAKE II is also able to pick up an assortment of weapons and ammunition, besides the default (and rather weak) blaster the player starts with. With a more powerful weapon, more damage is able to be inflicted on an opposing player more quickly. If a player dies, he/she drops all weaponry collected until that point and starts again in one of several 'spawn' points around the map.

1.2.2 Agents in Games

Agent

The term 'agent' in coming from the Latin *agere* ('to do') can be seen in general as merely something that acts (Russell and Norvig, 2003). With respect to AI, the term is generally representative of the software created (and hardware for robots) that is more than a program but less than a being. The term here, as was presented in the first paragraph of this chapter, can simply be seen as a simulated, automated game player. Agents in the computer games industry are generally known as 'NPCs' (Non-player characters) or 'bots'.

General Agent

A 'general' agent in this thesis is seen as an agent that has multi-taskability, that is, able to perform most or all tasks required within a computer game setting. This is in contrast to 'general game players' that are agents that are able to play a number of different (board) games. While this definition does not encapsulate Newell and Simon's 'general intelligence' (i.e. a human-level of intelligence) it has implications towards that effect; an agent who is able to manage successfully with the complexity of environments and tasks in modern computer games is believed to require a broad range of intelligent behaviours.

Complete Agent

A 'whole' or 'complete' agent implies a general agent with sensors and effectors to sense and interact with the environment. A complete agent would be both situated and embodied.

Situatedness

Russell and Norvig (2003) relate the of situatedness to the idea of an agent being embedded (i.e. immersed) within an environment, having to deal with continuous sensory input. In relation to this research, situatedness can be recognised as the idea of the agent being modelled interacting directly with the simulated computer game environment that surrounds it. While the agent might have abstract reasoning/cognition systems, a means to take in continuous sensory data and a means to use effectors on the environment is important. This is in contrast to the behaviour-based robotics (or 'nouvelle AI') idea of situatedness, which requires the agent to be in the real world, not dealing with abstract descriptions at all (Brooks, 1991). Like robots in the real world, these agents still require whole agent architectures including sensors and motor (effector) systems (see Figure 1.2 for a very basic agent architecture), but because the environment is simulated, issues like noise in sensory data do not occur, meaning greater concentration on the agent's cognitive ability is possible. In contrast, a non-situated agent is one that is separated from the environment it interacts with, with some form of *interface* that translates (lower-level) environmental data back and forth to (higher-level) abstract notions the agent can reason with (Berndt et al., 2005) (see Figure 1.3).

Embodiment

The concept of embodiment used in this thesis is similar to the one used in nouvelle AI. The agents experience the world through bodies (Brooks, 1991), though as before 'the world' relates to the simulated environment the agent is based in. An embodied agent's actions affect the world, while a disembodied agent uses some means of dialogue to affect actions through other agents.

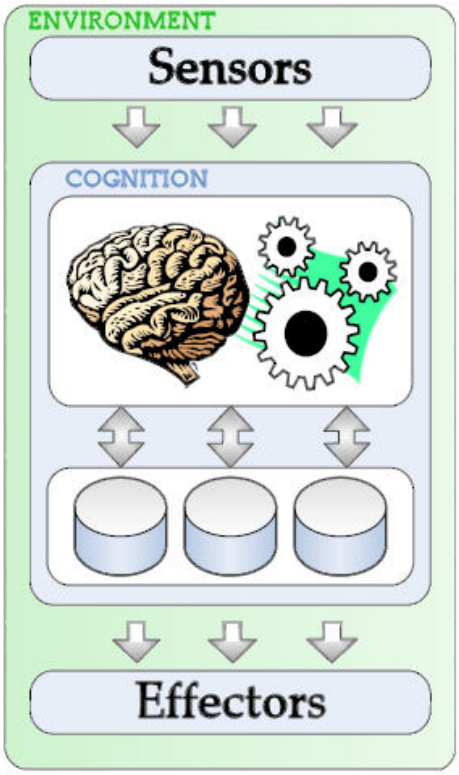

Figure 1.2: Basic Architecture of a Situated Agent: The agents sensors are directly connected to its cognition layer (seen here with prior knowledge content to draw upon) which in turn activates effectors to interact with the environment the entire agent is embedded within.

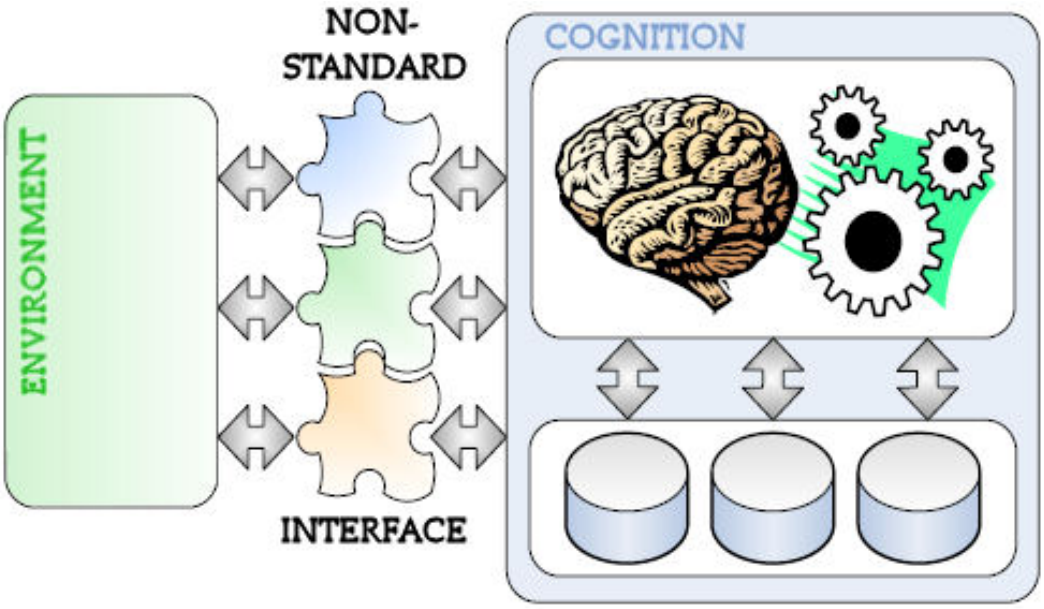

Figure 1.3: Non-Situated Agent with Non-Standard Interface: The agent gets information/acts on the environment only through an interface

Symbolic/Sub-symbolic Agents

While it is dangerous to subsume the philosophical minefield that is artificial intelligence in any fashion, for the purposes of this thesis it is useful to cut a rough line between symbolic and sub-symbolic agents. The term symbolic agent implies an agent which reasons using largely abstract or qualitative representations of the world it deals with. Symbolic agents have their roots in 'traditional' AI methodologies (e.g. Newell and Simon's 'Physical Symbol System Hypothesis') and are often deliberative in nature. Qualitative descriptions of continuous data are more human-understandable, do not require equations and in general mean the world can be described with less data than quantitative methods (Forbus, 1997). Symbolic agents are often non-situated, receiving transduced information through an interface or expert and requiring an internal world model to reason with. However, with research into qualitative representations of worlds this is changing, with the use of qualitative spatial reasoning (Freksa, 1991; Forbus, 1997; Guesgen, 2002; Forbus et al., 2002), a field that uses abstractions to model physical space, and qualitative physics (Forbus, 1997), a field that attempts to model the physics of a world using rules rather than exact quantitative methods. Sub-symbolic agents are often situated and reactive in nature, dealing directly with the quantitative data received from the surrounding environment. One of the strongest

advocates of sub-symbolic approaches has been Brooks with his 'subsumption' architecture (Brooks, 1986). Agents combining both symbolic and sub-symbolic components can be considered 'hybrid' agents.

Cognitive Architectures

Agents performing complex tasks often require structuring of cognitive components at a *systems* level (Laird et al., 1987) to work, in turn or in concert, to be able to reason, make decisions and effect changes on the environment. The cognition component shown in Figures 1.2 and 1.3 can be seen as a basic cognitive architecture that has a reasoning module and a background knowledge module to draw from.

Examples of Symbolic Cognitive Architectures A more complex example of a cognitive architecture is 'Soar' (Laird et al., 1987), built to model human cognition, which has quite famously seen a game agent implementation in 'Quakebot' (Laird and van Lent, 1999; Laird and Duchi, 2000). The original Soar cognitive architecture included procedural (long and short term) memory modules, a learning process called "chunking", a decision procedure module, as well as symbolic perception/action input/output. Soar 9 Laird (2008), the latest incarnation, includes reinforcement, semantic and episodic learners, procedural, episodic and semantic long term memories as well as an extension to handle sub-symbolic information. Other cognitive architectures of note are 'ICARUS' (Langley et al., 1991) and 'ACT-R' (Anderson and Lebiere, 1998).

Example of a Sub-Symbolic Cognitive Architectures An example of a cognitive architecture for a sub-symbolic agent is Thurau et al's three tier agent (Thurau et al., 2004c), described in detail in section 3.3.1. The basic structure is a combination of what was seen as the three levels of cognition: reactive, tactical and strategic. While a complete three tier agent was never completed, the group did implement an agent incorporating reinforcement and bayesian learning components integrating strategic and reactive levels, which was able to move around a map in a human-like fashion (see section 3.3.2).

Motivations for Cognitive Architectures

These complex cognitive architectures are often linked with attempts to model 'whole' or 'complete' agents that are both situated and embodied. Efficiency concerns are

secondary to implementing as 'complete' an agent brain as possible within the limits of the FPS environment the agent is situated in.

Agent Learning

A learning agent contains both a performance and learning element (Russell and Norvig, 2003), the performance element determining the agent's actions and the learning element altering the performance element so the agent can make better decisions. Three major issues to regard in the design of a learning scheme are the components of the agent that are planned to be learnt (e.g. reactive level, strategic level, etc), the level of feedback the learner will receive (i.e. supervised, unsupervised, reinforcement) and the representation of the learned information (e.g. rule-based (propositional/first-order), neural nets etc.) (Russell and Norvig, 2003). The original conception of the learning to be provided from interactive game logs was the outline of an implementation of a strategic layered component, using a somewhat-supervised level of feedback, using a rule-based representation scheme. This is further outlined in Chapter 4, Design.

Supervised/Unsupervised Learning

In the strict sense of Russell and Norvig (2003)'s definition of respective feedback schemas, interactive game logs (without added trainer intervention) are a form of supervised learning, in that they contain examples of both inputs and outputs in conveying the complete behaviour of an expert player (see subsection 1.2.4). However, it must be recognised even for schemes using neural networks or case-based learning regimes, some level of trainer intervention in presenting data would be necessary (including elucidating causal relations, see section 6.2.1). In this respect, in this research it is the level of trainer intervention in presentation of this data that is indicative of the level of feedback (supervised/unsupervised).

1.2.3 Knowledge Representation

This section defines concepts used throughout this thesis with regards to the way knowledge is stored or represented.

Procedural/Declarative Knowledge

Complex knowledge can separated into procedural and declarative types: procedural knowledge is knowing *how*, that is, use-specific knowledge; declarative knowledge is knowing *that*, that is, factual knowledge (Chi and Ohlsson, 2005).

Explicit/Implicit Knowledge

Explicit knowledge is knowledge that can be expressed in concrete terms, such as rules. Tacit or implicit knowledge is knowledge that is unable to be expressed in concrete terms, such as the knowledge of how to ride a bike (see Reber (1996)).

Continuous/Static Data

The term continuous data in this thesis relates to the stored information of unbroken activity over a period of time. An example of a single piece of continuous data is the position of a person for an instant in time that is part of a series of positions that together determine that person's movement over a certain time period. By contrast, static data relates to discrete information that stands on its own, such as total values or overall percentages.

1.2.4 Interactive Computer Game Logs

There are large repositories of game logs that are freely accessible on the internet. These logs represent millions of hours of motivated and task-oriented behaviour by a considerable number of people. The logs range from the purely statistical to others that capture the network traffic between the game client and game server several times a second, which, integrated with the correct map and parsing tools can provide a researcher with every action a player has made over the entirety of a game. Subject to motivation, game logs have the potential to provide opportunities to a variety of research fields. However, this research will focus on their use in artificial intelligence (AI) application.

As with any source of information, it is not only the nature of the content, but also the difficulty in extracting useable information that is indicative of an overall level of usefulness; both concerns are addressed in this thesis.

Composition of Interactive Game Logs

A computer game log can be recognised as a file that is used to store captured (usually live) game data, often in a sequential or time-stamped manner. A game log file can be human text readable, or require a parsing application (for an example see Gorman et al. (2005)). Whether it is the sequential moves taken in a turn based game or a FPS 'demo' file that captures all server-client network traffic down to time frames of a tenth of a second, the defining and encompassing attribute of the term's use in this thesis is the storage of game activity in an abridged manner.

The data in statistical logs might be lists of summed action or behavioural attributes (i.e number of kills made, aiming percentage). In demo logs the data includes every movement made by the player and other entities close to the player (opponents, bullets, explosions) to a fine grained level, including human ticks such as moments of indecisiveness or jumping unnecessarily off the top of a lift. The behaviour captured within these logs is generally of an expert level, with sequences of actions that can lead to complex manoeuvres.

Game Log Repositories

Game log repositories are large online databases containing thousands of game log files generally for the use of game enthusiasts. A repository may double as a means of providing parsing and statistical calculation of attributes (such as aiming accuracy for instance). An example of such a repository is 'The Blarghalyzer' (Blargh, 2005), a repository for Team Fortress Classic game logs. A more common type of repository is in the form of the collection of demos for video game log viewing enthusiasts. An example of this kind of repository is 'Challenge TV' (dethkultur et al., 2000). The first kind of repository provides a means for game enthusiasts to judge and improve their game performance, but can mean a loss of the sequential nature and granularity of the data within the logs. With the size of these repositories, this still means large amounts of data could be useful for many research endeavours, however it poses limitations for agent modelling. In particular, *tabula rasa* approaches (i.e. 'from scratch' learning; bootstrapping) to agent modeling require a differentiable, sequential and fine grained level of information. In contrast, whilst perhaps trivial to complete agent modeling, basic statistical level logs would be good for a game developer, hoping to increase the challenge level of NPCs as the player improved his gameplay, by comparing aiming accuracy statistics.

1.3 Purpose of the study

This section looks further at the motivations and aims of this research, already partly alluded to in the previous sections of this introduction.

1.3.1 Motivation

The main motivations for the investigation of using downloaded game logs to outline a method of agent implementation lie within the challenges and motivations inherent in the AI industry, namely to answer the question of whether machines can think, and the offshoots and opportunities for humanity in the AI research community trying to answer this question. While this research does not lay claim to any major breakthrough, it is thought through the investigation into an area that has not seen much attention, useful conclusions have been drawn to be of use to future researchers in the area. This section outlines the motivation for choosing downloaded game logs to work with, choosing computer games as a test-bed for research, the particular choice of the FPS genre, the use of computer game logs for agent research and the choice of rule-based data mining methods.

On the Problems with Supervised Learning Methods

Creating and elucidating specific actions to be learnt by an agent can be time-consuming and labour-intensive. A means for automating this process as much as possible would be helpful in a number of domains. It is believed that the ready-made downloadable game logs could be a means to fill this niche.

On the current failings of Computer Game Agents

Current computer game agents, though able to perform many if not all tasks to a competent level, are quickly tired of by players who migrate to multi-player versions of the game (Laird and van Lent, 2000). These agents generally do not include a learning component so weaknesses in their play can be taken advantage of. Besides this, the agents themselves seem to lack a human-like presence, with evidence of their artificiality all too evident: Sengers has noted:

> "predominant AI approaches to modeling agents result in behavior that is fragmented, depersonalized, lifeless, and incomprehensible" Sengers (2002, pg. 95).

Because the level of behaviour in downloaded game logs is expert-level and the granularity of the data captured in some logs is very fine, in a similar way to Thurau et al's motion-modelling (Thurau et al., 2005) (see section 3.3.2), the human ticks and hesitancies caught by these game logs could provide behaviours to agents to make them seem more human.

On the choice of using computer games for research

As has already been alluded to previously (see section 1.1.2), computer games provide a wealth of opportunity for academic AI researchers. These reasons include the complex and increasingly realistic environments, human-interactivity and open source code. Also of use in testing such agents are the embedded AI components and avatars that can be easily built upon and the goal-oriented, task specific behaviour often required merely to survive in these environments. Realistic simulated environments are a boon to researchers in providing a means of approximating the real world without the sensory and vision issues associated with robotic agent approaches (Laird and van Lent, 2000).

In addition, Laird and van Lent (2000) believed the economic driving force of the industry would serve to increase the usefulness of computer games to the academic community as competition and economics of scale would fuel their complexity, power and interactivity. As a means of further proof, Crandall and Sidak (2006) has estimated that since games revenue has been increasing every year by 15%, by 2009 sales of entertainment software in the US alone will top US$15 billion.

On the choice of FPS Shooters

The First Person Shooter (FPS) genre has been popular with researchers for its complexities (three dimensional environments, real-time decision making requirements) and its availability (a number of game development companies have released their source code over the internet for user generated content and research (id Software Inc., 1997; Valve Software, 1998)). This has meant there is a rich store of research and tools to build upon. In addition, the genre of game was also determined by the game log repositories found; FPS log repositories are by far the most prevalent.

On the choice of Downloaded Computer Game Logs for research

The number of internet sites that house large, freely accessible game log repositories, built using game logs uploaded by game enthusiasts, point to a wealth of highly motivated, goal-oriented, task specific behavioural data. With the size of these repositories, it is not extreme to imagine that even the sparsely populated, generally statistical game logs might provide a depth of information that, once mined to find patterns of behaviour, could be directly applied to agent learning. For the downloaded demo game logs, this is even more conceivable, as the log contents consist of fine motor interactive controls of enthused players in well defined, but complex, simulated environments down to split second granularity. It is also believed that while a number of different agent implementations using a variety of learning schemes has been seen (see chapter 3), learning from computer game logs is an area that has seen much less research, in particular using symbolic/rule-based methods.

On the use of Rule-Based Data Mining Methods

Agents using rules have a greater ability to reason and incorporate prior knowledge. FPS player actions consist of a broad skill set requiring strategic, tactical and reactive behaviours. With the success of a previous rule-based agent in this environment (Laird and van Lent, 1999) and mindful of the quantity of data to be faced with, it was thought that a data mining approach incorporating rule learning would most effectively point to a means of agent implementation.

As will be seen from section 3.3, there have been several approaches taken to modelling agents using the data from game logs using sub-symbolic methods. It was thought a symbolic approach to the problem might bring some new ideas to light.

1.3.2 Aims of the Research

The aim of this research is to investigate the hypothesis that downloaded, expert-level, interactive computer game logs could be a source for general agent learning by supplementing or even replacing supervised learning methods. This hypothesis raises a number of issues that are aimed to be addressed in this thesis:

1. Are downloaded interactive computer game logs useful for agent modelling research?

2. What are the benefits of using these game logs?

3. What are the issues related with using downloaded computer game logs?
4. Is there usable knowledge for agent learning within these logs?
5. What form of knowledge is the behaviours encapsulated within the game log data?
6. Is this knowledge extractable, and if not, why not?
7. Is it possible to bring this knowledge to agent implementation, and if not, why not?
8. If it is not possible, what means are there to solve these issues?

Chapter 2

Rule-Based Data Mining

Rules are one of the most human readable and expressive forms of presenting a learned hypotheses from a set of data Mitchell (1997). That is, their innate strengths lie in their ability to define patterns in the data in a form that is uniquely understandable for people, and an ability to express these patterns in a form that effectively highlights the relationships within these patterns. This section outlines the rule-based methods used in this research.

2.1 Classification Rules

Classification rules are generally used as a set of rules to subsume data into separate classes as a means of prediction for future data. This subsection covers the general sequential algorithm common to most classifiers, an introduction to an industrial strength classification rule learner in RIPPER, a decision tree rule classification learner and finally a look at a massive search learner called Brute.

2.1.1 Sequential Covering Algorithms

A sequential covering algorithm is a means of simplifying the task of classifying data sets by breaking the task into a succession of smaller problems, each of which only need one rule to be learnt (Mitchell, 1997). The process can be broken down into three steps:

1. Learn a rule using greedy search
2. Remove instances from the data set the rule correctly predicts
3. Repeat procedure until desired fraction of data set is predicted

This procedure works with discrete attributes. However, it is possible to use numeric attributes if they are grouped into less-than/greater-than discrete sets. An example of a widely used algorithm that uses the basic sequential covering process outlined above is RIPPER (Repeated Incremental Pruning to Produce Error Reduction), outlined next.

2.1.2 Ripper Rule Learner

The RIPPER rule learner (Cohen, 1995) is an industrial strength rule classifier that uses a repeated incremental reduced-error pruning process with the basic sequential covering algorithm outlined above with some added optimisation, mop-up and clean-up stages to find classification rules in data.

The incremental pruning process begins after splitting the data set into training and testing sets by splitting the training set again into 'growing' and 'pruning' sets. The growing set is used to build a rule set for each attribute the data is being sorted by (the 'class' attribute) incrementally by collecting rules built by adding tests until each rule has perfect accuracy.

The pruning set is then used to test the new rule by taking away tests from the rule to see if truncated versions of the rule are better. The sets are separate to avoid overfitting of the rules. A drawback to this is that important rules that might only be learned from instances in the pruning set could be missed, or being only a third of the training data, the pruning set could be misrepresentative of the entire training set. These problems can be addressed by resplitting the training data at the end of each cycle of the incremental reduced-error pruning process, after the pruning stage is ended by adding the discovered rule to the set of new rules and removing the instances it covered within the data set, as per the sequential covering algorithm. A post-pruning strategy is preferred because it is possible that a very good rule might be found that could be missed by a aggressive pre-pruning strategy (Witten and Frank, 2005). Each cycle of the incremental pruning algorithm is stopped when either:

1. all instances have been covered
2. the error rate of the last found rule was less than an arbitrary value of 50%
3. or when the total description length of the rule set and examples is greater than the smallest description length obtained by a certain number of bits (Cohen (1995) used 64)

Once a rule set has been collected for a particular class, it goes through an optimisation process which tests each rule with two rule variants (produced at this stage of the algorithm also by reduced-error pruning using a new split of the training set where all instances covered by every other rule in the set is removed). If one of the variant rules produces a lower description length of the rule set it replaces the original rule. Next, there is a small mop-up stage that creates rules for any left over instances, followed finally by a clean-up stage that ensures each of the rules produced in the rule set do not increase the overall description length. The entire process is then repeated for the next largest class attribute (Witten and Frank, 2005).

2.1.3 Decision Tree Rule Learning

Using each of the leaves of a decision tree as the consequents of the rule (second half of the rule) and the conjunction of all the tests encountered on the path from the root to each of the leaves as the antecedent (first half of the rule) is one way rules can be created from decision trees, though this simple process usually produces rules that are overly complex.

Another way is to use the sequential covering algorithm as outlined above while using the leaf with the largest rule coverage of a built-from-scratch, pruned, partial decision tree to create each rule.

Partial trees are created by first choosing a test using an information gain heuristic that separates the training set into subsets (new nodes below the root node of the tree). These subsets are then expanded in order of their entropy (information value based on the number of positive and negative attributes for that attribute) as a subset with a low entropy is less likely to have a large sub-tree, so is more likely produce a more generalised rule. This process is continued until a leaf is reached, then continued using back-tracking. Once a node has all its children expanded as leaves, if the estimated error rate of that expanded node is greater than that of a single leaf, the expanded node is pruned and replaced by the leaf. Backtracking continues the process until a node is reached that has no children expanded, where the process ends (Witten and Frank, 2005).

2.1.4 Criteria for Choosing Rule Tests

There are several means to measure the effectiveness of the addition of a test to the overall 'worth' of a rule (Witten and Frank, 2005). Basically, this worth can be seen in

the rule covering as many positive instances as possible, while minimising the negative instances it covers. One means is ratio maximisation, which tries to maximise rule accuracy:

$$p/t$$

(where p is the number of positive instances the rule covers, t is the total number of instances covered by the rule).

A flaw in the ratio maximisation measure can be seen with rules covering only positive instances, even of a very small number, will be preferred even over rules that cover a very high percentage of a high number of instances (e.g. a rule covering 1 positive instance will be preferred to a rule covering 1000/1001 instances) (Witten and Frank, 2005). A second means is information gain, which tries to represent the total information gained in adding the latest test in comparison to the old rule:

$$p[log(p/t) - log(P/T)]$$

(where as before, p and t are respectively is the number of positive and total instances the rule covers, while P and T are respectively the number of positive and total instances the rule covered before the new test was added) (Witten and Frank, 2005).

The information gain measure places more emphasis on the number of positive examples the rule covers than the percentage of instances that are positive. This means as tests are added to each rule to produce perfect accuracy, algorithms using the ratio maximisation will generally finish faster, eliminating outlying instances early on, sometimes meaning a simpler decision making process in discovering a more general rule. The information gain measure creates rules with high coverage first, leaving special cases for later. It is hard to compare which strategy is better, especially since this stage will generally lead into a pruning stage, which often has its own means for measuring the worth of a rule. A simple measure at the pruning stage is to see how well each rule on its own (out of the entire set generated to predict the class) predicts the class it is part of predicting from other classes (Witten and Frank, 2005):

$$[p + (N - n)]/T$$

where p is the positive instances out of t total instances the rule covers, and P is the total number of instances of this class in the total T number of instances. n = t-p and N = T-P. It can be seen that this measure would prefer a rule that achieved a

coverage of 2000 positive examples from a total of 3000 instances it originally covered (i.e. n=1000) to a rule that achieved a coverage of 1000 positive examples from a total of 1001 (i.e. n=1). In the first the measure is [1000+N]/T and in the second the measure is [999+T]/N Witten and Frank (2005). Sometimes, the ratio maximisation measure is used here instead, however as has been mentioned, it suffers its own issues.

2.1.5 Brute

Brute (Riddle et al., 1994) is a rule classifier that uses an efficient massive search algorithm to run through the entire set of possible rules to a user specified search depth.

Efficiency is an important part of the Brute algorithm, so it uses a number of pruning steps to ignore sections of the overall search space that cannot possibly contain useful rules. This reduces the overall search space and run time of the program, often by a factor of 1000 (Segal and Etzioni, 1997).

2.2 Associative Rules

Association or associative rules are used to predict any attribute of the data set, plus combinations of attributes as well. Unlike rule classifiers, association rule methods do not create rules that are disjunctive of each other or rules that are to be used as sets. Instead, association rules are independent of each other and are indicative of an assortment of patterns within the data (Witten and Frank, 2005).

Because testing combinations of attributes is computationally expensive, association rule finding methods concentrate on those rules whose coverage (the number of correctly predicted instances the rule makes) is high (as the minimum coverage required by the user is decreased, the more computationally expensive the association rule process becomes). This is done by collecting and building combinations of attributes (item sets) that have coverage of a specified minimum number of instances. From these item sets, rules are generated and discarded if not over a prespecified accuracy (proportion of coverage that the rule predicts correctly) (Witten and Frank, 2005).

Because of the computational expense, association rule learners are generally used when instances have binary and sparse attributes (Witten and Frank, 2005).

2.2.1 Temporal Association Rules

Temporal association mining is the process of discovering hidden relationships between sequences of events (Antunes and Oliveira, 2001). Many temporal data mining process require the events to be symbolic and discrete (i.e. a *temporal sequence*); if the representation is continuous and qualitative (i.e. a *time series*), a pre-processing stage is required. This is usually done with some form of discretisation. A contentious issue in temporal data mining that has links to this research is the developing of algorithms to label the discretised values of a time series data set without having partitioned the data automatically (Daw et al., 2003). This often is the case when the time series data is particularly obtuse or noisy, or the researcher him/herself does not know the underlying domain rules (i.e. domain physics). Automated discretisation methods include clustering (Das et al., 1998) or comparing the shapes of sequences to set shapes that are labelled with symbols of an alphabet (Agrawal et al., 1995). Once this is achieved, an associative rule learner, such as variants of the *apriori* method is used (Agrawal and Srikant, 1995).

2.2.2 Relational Rule Mining

Relational rule mining can be seen as a first order extension of the rule miners outlined above which are propositional in nature. First order rules are more expressive than propositional rules (Mitchell, 1997) and the rules they create are more intuitive and provide a more concise description of the concept by indicating relationships between attributes (Witten and Frank, 2005). The data is required to be set out in database tables with relationship links, from which an algorithm can search for relational patterns. These algorithms are generally variants on algorithms of the field of the inductive learning of relational rules, inductive logic programming (ILP) (Muggleton, 1995).

The field of ILP is the source of most relational data mining algorithms (Lavrac and Dzeroski, 1994). An first order extension of a propositional classification miner is FOIL (Quinlan, 1990), while a first order extension of an associative miner is WARMR (King et al., 2001).

2.3 Overfitting

Overfitting is caused when the rules that define one set of data particularly well do not generalise well to new data. This can be minimised as mentioned above by learning rules using only two-thirds of the data (training set) and evaluating the found rules on the remaining data (test set). Another means is by ten-fold cross-validation which trains the data on 9/10ths of a data set then evaluates it on the final 1/10th ten times. For rule learners such as Brute, a chi-squared test is required. This involves an arduous method of randomising a column of data and finding rules on that new data a number of times ($\sim$100 is good). By collecting these rules and applying them all to the original data set (with the original rule to be tested) the original rule can be deemed non-random if it performs within the top 10-20% of all rules.

Chapter 3

Literature Review

The following examples of research modelling agents in computer game environments have been subsumed into symbolic (often declarative/inference/top-down) and sub-symbolic (procedural/reactive/bottom-up) approaches. Generally, symbolic examples found have focussed on utilising game environments to implement practical examples of Newell's 'Unified theory of Cognition' (Newell, 1990)–centralised, hierarchical task (implemented largely as hand coded rules) based cognitive architectures that generate rather than emulate human-like behaviour. Lower-level symbolic directions are hard coded to procedural actions to make these symbolic agents embodied and situated. In contrast, the sub-symbolic approach examples listed after the symbolic examples deal directly with the FPS environment, but only display reactive behaviours. The sub-symbolic approaches to modelling agent behaviour in computer games shown below have their basis in robotics and behaviour-based architectures (i.e generally using methods from Brooks (1986)). Although a generally symbolic approach was taken in this research, sub-symbolic papers are of interest, not only for their comparative value to the symbolic examples but also for their use in future discussion. Also of interest is their dichotomous theoretic basis: one of the fundamental aspects in the founding of the behaviour-based or 'nouvelle AI' field was the physical grounding of the agent in the real world (Brooks, 1990, 1999). While this 'fundamentalist' view of sub-symbolic agent modelling does not completely dominate the field (Pfeifer and Scheier (2001) in particular call for more research in simulated environments), it is a strongly held view: Brooks (1999) ends with "Don't use simulation as your primary testbed. In the long run you will be wasting your time and your sponsor's money".

This review will first look at examples of research with computer agents as a means of introduction to the field. From there, a move closer to the subject at hand will be made, looking at research with symbolic and sub-symbolic agents using game logs to

learn behaviour from. Finally, as something of an aside, an introduction to two cases of data mining with musical performance representation data that have strong analogies with this research will be included here and used in further discussion in later chapters.

3.1 Research with General Agents in Computer Game Environments

This section will look firstly at a number of general agents. Firstly, embodied agents in computer games (both symbolic and sub-symbolic) with examples from different cognitive architectures. Secondly, a look at a few general disembodied agents that have a greater relevance to this research as they are based on examples of expert play. A look at agent interfaces will be included here, with their relevance to non-situated agents.

3.1.1 Research with Embodied General Agents

Pengi

One of the best known examples of research modelling an agent in a computer game is the completely reactive agent 'Pengi' (Agre and Chapman, 1987). Pengi's agent architecture consisted of a single central system for cognition and several peripheral systems for sensor and effector control. The agent was landmark in that it was one of the first examples of researchers shying away from a traditional declarative planning approach. Pengi used a combinatorial gated network approach using 'aspects' instead of variables bound to symbols. Aspects were described by the authors as "the relevant properties of the immediate situation" (Agre and Chapman, 1987). More specifically, they can be seen as propositional representations of objects in the world, that instead of being given arbitrary labels, are identified by the function they fill in ongoing activity (Finney et al., 2002).

Soar Cognitive Architecture

A notable example of research using an agent in a more modern, three-dimensional setting is from Laird and van Lent (1999) and Laird (2001), who show a FPS agent implementation of the Soar cognitive architecture in QUAKE II called 'Quakebot'. Quakebot is able to build its own map of a level being explored and to display some

anticipatory attributes, such as setting ambushes by using its own knowledge to decide what the enemy will do next. Quakebot's internal configuration is based around an operator hierarchy which uses over 700 hand coded rules as well as a number of numeric parameters.

An agent implementation of the Soar architecture sharing many of the same characteristics of Quakebot is being developed (Wray et al., 2005) by the Office of Naval Research's Virtual Training Environment (VIRTE) program. Previous work on this project (Best and Lebiere, 2003; Best et al., 2002) saw implementation of agents using the 'ACT-R' cognitive architecture (Anderson and Lebiere, 1998).

Another example of the Soar cognitive architecture being used in agent modelling is by Konik and Laird (2004) who use a first-person perspective story-telling game built using the FPS environment of 'Unreal Tournament' (Epic Games and Digital Extremes, 1999) as a "supervised concept learning setting" where agents learn first-order rules for procedural goal hierarchies by observing expert decisions. The decisions are in the form of highly structured operators and annotated by the expert as they play the game. Rules are learned by an inductive logic programming algorithm (using inverse entailment (Muggleton, 1995)) from positive and negative examples tried by the agent and stored in an episodic memory.

Icarus Cognitive Architecture

Returning to urban combat settings, Choi et al. (2007) use another cognitive architecture, ICARUS, to model an agent in the game 'Urban Combat', a modification of 'Quake III Arena' (id Software Inc., 1999). Their approach is similar to Quakebot; the agent's actions are largely hand-coded with an explanation based learning scheme that creates specific rules by simplifying a list of actions performed by hand coded exploration skills. Additional attributes of the agent include the ability to learn routes in new environments and reactive behaviour in the midst of goal directed activity. While the basic premise of ICARUS is the same as Soar (each is based on Newell's Unified Theory of Cognition (Newell, 1990)), there are a few structural differences that ICARUS possesses including disparate concept, skill and goal memories, hierarchically arranged long-term memories and procedural tasks indexed by the goals they achieve (Choi et al., 2007).

Ledgewalker

An example of modelling a general FPS agent using subsumption architecture techniques is from Khoo et al. (2002), who's reactive agent 'Ledgewalker' used combinations of finite state machines that were already a feature of many of many agents in the computer games industry. In comparison to Quakebot, the agent was much more efficient, not requiring its own CPU to run, but was unable to reason or plan about its circumstances, making it purely reactive in nature.

3.1.2 Research with Disembodied General Agents

Mayor

Fasciano (1996) implemented a case-based reasoning disembodied, non-situated agent to play the real-time strategy game SimCity. The agent was required to be able to handle short and long-term tasks, interruptions to those tasks with city emergencies, having too many things to do (being able to prioritise), plus not knowing exactly how game representations affect each other (e.g. it is not obvious how property values are affected by crime) (Fasciano, 1996). The agent was able to do these tasks by separating in-game tasks to specific declarative and reactive modules for each area of concern in the game, which when necessary post to a task agenda list. The task agenda list is prioritised centrally and tasks are allocated accordingly.

CIGARS

Louis and Miles (2005) used Case-Injected Genetic Algorithms ('CIGARS') to learn how to play a three-dimensional real-time strategy computer game effectively. The genetic algorithm part of the agent provided an adaptive learning component to the agent, while the case-bases were used to speed up this process (i.e. the genetic algorithm was not completely randomised).

3.1.3 Interfaces

Interfaces can be seen as a means for an agent to interact with the computer game environment without having to be immersed inside it. For agents like Mayor and CIGARS, it is tempting to class them as non-situated agents as they work in games where a graphical user interface (GUI) separates the player from being directly involved in the

action. However, though disembodied, and unlike agents such as Pengi, they do not use the visual representations given by the player GUI but the underlying data instead. In this way they have sensors and effectors of sorts that effectively immerse them in the environment they are in. Academic researchers look to games for the immersive capabilities of their environments as well as the situated agent movement as a defining aspect of intelligence (see especially Brooks (1991)). Due to a lack of an overarching general AI game interface (Berndt et al., 2005) non-situated agents are not the norm in this field. There are some examples of agent interfaces, for instance, Adobbati et al. (2001) built a general application interface ('Gamebots'): a research platform of modifiable environments and agents using the 'Unreal Tournament' (Epic Games and Digital Extremes, 1999) game engine. This platform has subsequently been used to create a number of research projects, including 'UnrealTriage' (McGrath and Hill, 2004): a mass casualty incident simulator for training of emergency response triaging; and Fielding et al's (Fielding et al., 2004) Embodied Reporting Agents: participatory yet passive agents that generate reports of game action they 'see' for the benefit of non participatory players. At the time of publishing their paper, Adobbati et al. (2001) were looking at creating analytical tools for "calculating further statistics from the log, like bot aiming accuracy or average bot speed, to aid in the evaluation of bot and team behavior" (Adobbati et al., 2001), an early pointer towards agent modelling using game logs.

For the later half of this research, the Quake II Agent Simulation Environment Application Programming Interface (QASE API) (Gorman et al., 2005) was used, which includes several basic bot frames, inbuilt neural network and genetic algorithm generators, a map waypoint generator, Matlab and a game log file parser. Other APIs include the FEAR SDK (Champandard, 2002) and TIELT (Molineaux and Aha, 2005).

3.2 Research with Robocup Agents, Game Logs and Data Mining

The Robocup competition has seen quite a few examples of modelling agents by using data mining techniques with game logs. In this section several examples of such research are presented. Firstly is a look at the Robocup coach competition, followed by a look at the use of imitative learning processes to implement case-based agents in the Robocup Simulation League. Agents with larger symbolic compositions follow this, with another

case-based example that uses sequences of team actions to make a Robocup soccer team less reactive, at the expense of losing some automaticity in the learning process (Floyd et al., 2008). Finally, an example of rule-based learning techniques from Robocup agent logs is presented. An overriding differentiation to the current research in all of the examples outlined below is the general level of complexity of data within the game logs themselves, due to the relatively simple nature of the agents, game rules and two-dimensional environment the Robocup games are situated in.

Robocup Coach Competition

One of the best known uses of game logs to model the behaviour of simulated agents is the Robocup Coach Competition. The competition uses game logs of standardised opponent teams to model a disembodied agent, which is scored by the pattern prediction it makes about the opponent its own team plays as well as its ability to send out instructions to its soccer playing agents at specific intervals. Kuhlmann et al. (2006) describe a successful implementation of such a coach agent, which competed at the 2005 Robocup. Research examples from the Robocup Coach Competition can be differentiated from the current research in that it is mainly interested in opponent team modelling, rather than single player modelling.

Robocup Imitation of Individual Agents

The Robocup Simulation League has had similar examples of agent modelling, but using game logs that follow the behaviour of individual agents from their own point of view (c.f. the global field view of the coach agent game logs). One such example is from Floyd et al. (2008) who used a Case-Based Reasoning (CBR) approach to imitate other simulated players in a largely automated process. The agents created were stateless, single-layered and largely reactive, but were able to reproduce many of the behaviours of more complex agents. Cases were stored in a 'spatial knowledge representation format' which took the view and actions of an agent at discrete time instances (Floyd et al., 2008). If a playing agent is in a situation (i.e. has ball and opposing/team agents in similar positions) that is comparable to a stored case, it can use the action associated with that situation. This form of behavioural modelling is largely analogous to the needs of research presented in this paper, for unlike the Robocup Coach Competition, the content of the logs used are a direct reflection of the behaviour that is to be learnt

and the process of learning is largely automated. However a rule-based approach was explored to judge if a cognitive approach is possible.

Robocup Imitation of Team Gameplays

Ros et al. (2007) also used a case-based reasoning approach for modelling in a Robocup environment, but modelled sequences of play between agents. Cases were stored again from a single agent's point of view, here a 'reference' agent (like a captain in soccer team), but also included a sequence of actions each team mate should perform should that case be chosen. Unlike Floyd et al. (2008), the process was not automated, with knowledge representations being of a qualitative nature to take into account an uncertainty factor and to more easily describe the player positions with respect to the ball. The closest robot to the ball is chosen as the reference robot, who retrieves the most similar case and informs the other team mates which case has been chosen. Once each team mate has moved to the position it should be in, the sequence of actions for each of them to execute begins. If any of the robots recognises that the case is no longer applicable, a sequence of actions can be abandoned. A team modelling approach was abandoned with the TFC logs (see section 4.2.3). As it will be discussed later, a level of automaticity in the learning process with computer game logs is important (see section 5.3).

Sequential Pattern Mining

Lattner et al. (2005) uses association rule mining for pattern matching on qualitative representations mapped from the quantitative Robocup game log data. The process required two steps, firstly the mapping of quantitative to qualitative data and secondly the sequential pattern mining of that data. The Robocup logs provide quantitative data for every object (ball, opponents, team mates) in the form of each object's motion direction and speed and the spatial direction and distance for each pair of objects every tenth of a second. When these atttributes are taken as a sequence of events, segments of similar values can be made into discrete values using arbitrary threshold values (e.g. '*very close*' < 2; $2 <$ '*close*' < 4; $4 <$ '*far*' < 8 etc) in comparison to the segment's sequence of events average value. Similarly, predicates such as 'approaching' and 'departing' can be made by using a monotonicity-based segmentation method. Once a scene is represented symbolically, a pattern matching algorithm using a sliding window is implemented to generate rules (Lattner et al., 2005). The success this

research had with the less complex logs was hoped to be repeated with the more complex Quake II logs in the second part of this research (see section 4.2.4).

3.3 Learning from Game Log Data for Embodied Agents in Game Environments

This section goes into greater details of the papers being looked at as, although they are using largely sub-symbolic techniques and creating custom logs to suit their research, they have the most relevance to this research.

As has been seen in section 3.1.1, many sub-symbolic approaches to the modelling of 3D game agents struggled to create anything more than purely reactive agents. Behaviour-based agents based on Brooks' 'Subsumption Architecture' (Brooks, 1986) such as (Khoo et al., 2002) had no learning capabilities due to the difficulties in learning finite-state machines.

In comparision, Thurau et al. (2004c), in creating an imitative agent from QUAKE II game logs, proposed a three tier model of agent behaviour with strategic, tactical and reactive levels. This is probably due to their realisation that solely reactive approaches (Bauckhage et al., 2003) are insufficient to model the complex behaviours required for a realistic agent. While they produced separate examples of each tier and found some success in combining strategic and reactive levels (Thurau et al., 2003, 2004b,a), an agent using all three tiers was never implemented. Perhaps most successful was a later collaboration with Gorman and Humphrys (Bauckhage et al., 2007; Gorman et al., 2006a). This saw a combination of a strategic level behavioural implementation with a modified reactive level. The next section outlines how the three tier plan was to be implemented as a FPS game agent, an outline of this most recent agent with respect to its separate strategic and reactive levels and finally a look at two approaches that were taken to implement a tactical level which were never integrated into a complete agent.

3.3.1 Thurau et al's Three Tier Agent Behavioural Model

This section outlines Thurau et al's three tier behavioural model (Thurau et al., 2004c), originally outlined briefly in section 1.2.2. The model was based on Hollnagel's 'Contextual Control Model' (Hollnagel, 1993), a four tier psychological hierarchy of human be-

haviour, consisting of 'strategic' (planning ahead of present situation), 'tactical' (planning in current situation), 'opportunistic' (limited planning using environmental feedback) and 'scrambled' (no planning or very limited planning, including panic-reactive behaviour) levels (Hollnagel, 1993).

In comparison, the three tier behavioural model included a 'strategic' level to handle longer term goals, such as controlling important areas of the map and collecting health and weapon items (powerups), a 'tactical' level to deal with shorter term goals including ambush type behaviour and weapon control, and merged Hollnagel's 'opportunistic' and 'scrambled' levels into one 'reactive' level which dealt with aiming, jumping and firing - to be based on perceptual input.

This behaviour-based, simple tiered cognitive model can be contrasted to a examples of symbolic cognitive architectures which explicate the reasoning process as separate memory and learning modules. As has been mentioned, a complete agent using all three tiers was not implemented. However, it is of interest to look at last multi-tier agent to be implemented by this group (Gorman et al., 2006a) which used closely linked strategic and reactive levels to effect goal-driven behaviour and realistic movement modelling. This agent is described next by describing the strategic and reactive levels in more detail.

3.3.2 Gorman et al's Two Tier FPS Agent Research

Strategic Tier

Capturing Implicit Strategic Knowledge in Behaviour of Players at Item Pickup Points Gorman et al. (2006a)'s strategic level behaviour modelling was based on viewing the powerups that can be collected around the map (such as health, weapons and ammunition) as strategic goals, as well as viewing the agent's changing inventory (list of how many of each item the agent possesses at the current time - includes health) as strategic states. Essentially, it assumes the higher level strategic thinking that is approached explicitly as rules in symbolic methods will be seen (implicitly) in the behaviour of human players by the actions they choose to take when faced with a certain situation (low/high health, poor/good weapon acquired, low/high ammunition levels, etc). This assumption led to the group pulling away from a purely reactive model and will be discussed further in section 6.3.2).

Implementation of the Strategic Tier using a Markov Decision Process The points along the paths followed in these strategic inventory states to each strategic goal powerup were standardised by breaking down the complete set of observed human player positions from custom game logs they had created for the purpose, using clustering to form a set of topological nodes that represent typical player positions. In addition, observed human player transitions between these topological nodes were recorded in a two dimensional matrix, whose node-indexed members showed whether or not a player had been observed travelling between each pair of nodes that had been found. (i.e. for a matrix E, member $E_{ij} = 1$ if the player was noted to have moved between nodes i and j at some point). The combination of these topological nodes and transitions was viewed as a Markov Decision Process (MDP). The clustering performed in this part of the strategic modelling processcan be compared to the temporal rule mining clustering discussed in section 2.2.1.

Assigning MDP Utility Values for Reinforcement Learning of Paths A complete set of paths observed to be followed by the human player while in each inventory state was collected and provided a means of assigning rewards for this process to nodes: with each successive node on a path in a certain inventory state the reward was increased, to ensure the agent was more likely to stay on paths followed by players. Utility values for every node in every inventory state were learned by the agent with reinforcement learning using value iteration. To model a player's ability to gauge the importance of two or more items that need to be collected, a fuzzy clustering approach was used. Whenever the agent's inventory changes, a membership distribution across all clustered inventory states is calculated, comparing how similar the current inventory is to each of the clustered states. This membership function is used to calculate the final utility values.

Reactive Tier

Breaking Actions Down to Simple Movements to Build Sequences of Behaviour To construct action sequences to define the movements the agent makes between the topological nodes, Gorman et al. (2006a) modified Thurau et al's Bayesian approach (Thurau et al., 2005), which had been based on previous work using the same approach to model infant imitative learning for robots (Rao et al., 2004). Observed human players' movement vectors at every time frame of a recorded game were cate-

gorized using k-means clustering into a set of several hundred 'motor primitives', each consisting of a vector with a pitch, yaw, jump, weapon and firing component. Choice of each successive action is chosen deterministically by the action with the largest probability, worked out using Bayesian theorem, by considering the current node, the next node (chosen by the strategic navigation system using the calculated utility values) and the current inventory state. This process is one solution to the problem found in the current research in trying to break down continuous sequences of expert actions into their basic components, discussed in section 6.1.2.

Believeability Testing The motion created by this approach was considered by the authors to be very believable; in a 'believability test' of 'humaness' created by the authors (Gorman et al., 2006b), a panel of people with mixed gameplay expertise were asked to compare several twenty second, single-player gameplay videos showing the movements of this (imitative) agent, a rule-based agent or a human player. It was found that the imitative agent was misdiagnosed as the human player as much as the human player was correctly diagnosed as human, a proportion twice as high as the rule-based agent. This test for believability will be discussed further in section 6.3.2. The success of this process is testament to the fine grained level of data capture inherent to the demo log recording process, meaning intuitively human ticks and unconscious behaviours can be added to an agent's behaviours, giving it, as here, a closer representation of human players. This was included as one of the reasons interactive computer game logs are useful for agent learning research (see section 5.1.1) and is further discussed in section 6.1.1. Having visited two of the three levels of behaviours that Thurau et al. (2004c) outlined, we now turn to the last, the tactical level.

3.3.3 Sub-Symbolic Tactical Level Modelling Using Neural Networks

Tactical level modeling in FPSs is primarily concerned with the firing, aiming and handling of weapons, that is, when and where to shoot and what to shoot with. Extended tactical modeling deals with present situation planning and opponent modeling that can affect strategic level plans. Two different approaches to implementing a tactical agent using neural networks were tried by the same group that produced the agent described in the previous section. Both only implemented a basic tactical agent (no present situation planning, only weapon handling) that (as has been mentioned) was

never integrated with the strategic/reactive agent described in the previous section. The use of neural networks for tactical modelling can be compared to other possible imitative processes discussed in section 6.3.2. Neural networks though very good at learning non-linear functions can be difficult to integrate with other learning methods in hybrid agents (see section 6.3.3). Detail of the tactical modelling approaches is included as a means of comparison to this research's attempt at finding aiming rules.

Bauckhage and Thurau's Tactical Modelling Approach

Bauckhage and Thurau (2004) used Mixtures of Experts, a technique that uses the weighted sum of the outputs of a number of neural ('expert') networks (Jacobs et al., 1991). Weighting of the expert networks is achieved using a 'gating' network, another neural network connected to the output of each expert network, trained to favour different expert networks, given different situations. To simplify the difficulty of the problem, modeling was confined to two dimensions. Two configurations were tried, the first using three experts, all having separately learned the handling, aiming and firing of three separate guns from observed expert data, with a larger gating network to weight the sum of these experts' decisions. The second configuration had the expert networks learn a weapon each from the observed data and had the larger gating network choose the best gun depending on the situation faced. Both configurations were able to produce the behaviour seen in the observed data, both switching between weapons, aiming and firing in the appropriate circumstances.

Gorman and Humphrys' Tactical Modelling Approach

Gorman and Humphrys (2007) also used connected neural networks, but had one neural network each for aiming, firing and weapon handling. A three dimensional setting was used, but heavy data preparation and processing was required in the form of changing all observed positional global data into data relational to the agent itself. As an added feature to make the agent more human-like, the observed player's inaccuracy was included in the learning process. The agent was successful in learning aiming, firing and weapon handling behaviour, including leading behaviour (shooting in front of a moving enemy) with slower projectile weapons (such as the rocket launcher) as the amount of time available to the observed player increased. Some unusual behaviour was learned, some anticipated: occasionally the agent would frequently change his weapon in quick succession, a behaviour similar to humans scrolling through their available

weapons; and some not: the agent for some reason was able to track an enemy more easily when moving from right to left. It was worked out this was due to the greater ergonomic ease the observed human players had in moving their mouse in an inward arcing movement (right to left) as opposed to an outward sweep (left to right).

3.4 Data Mining using Complex Musical Performance Data

The two papers presented in this section are presented for their analogous nature to the current research and their use later on as a discussion tool for the mining of complex low level data.

Rule-Based Data Mining for Performance Music Data

Widmer (2003) presents an ensemble rule learning method 'PLCG' to find partial rule models in a set of expressive musical performance data (i.e data collected from measurements of performances of concert pianists). While the rule-mining algorithm was largely successful, and interesting performance principles were found, more abstract aspects of the music including phase and harmonic structure could not be explicitly found as rules. This finding has direct relevance to the research at hand and will be discussed further in section 6.2.1.

Case-Based Data Mining Techniques for Performance Music Data

In conrast, Lopez de Mantaras and Arcos (2002) presents a successful implementation of a computer system capable of performing expressive music using case-based reasoning techniques, relying on the implicit knowledge represented in the data rather than trying to make this knowledge explicit. Discussion relating to this issue can be seen in sections 6.2.1 and 6.3.2.

Chapter 4

Design and Implementation

This design and implementation chapter outlines the planning and approaches taken in investigating the use of downloaded, expert-level game logs for general agent learning. Section 4.1 presents the design of the research, including a review of the research hypothesis, conceptual outline and initial assumptions. Section 4.2 outlines the implementation taken, including the investigation of the game logs, the challenges faced, and attempts with the use of rule-based data methods to extract explicit knowledge.

4.1 Design

This section outlines the original hypothesis made, conception of an initial implementation and assumptions made.

4.1.1 Original Hypothesis

The original hypothesis of this research was that downloaded, expert-level, interactive computer game logs could be a source for general agent learning by supplementing or even replacing supervised learning methods. This hypothesis was based on motivations not only in the content and availability of the logs themselves (see section 1.2.4), but in the suitability of interactive computer games as a test-bed for general agents (see section 1.3.1).

4.1.2 Conception of Initial Implementation

The initial concept for an implementation of the hypothesis above was an investigation of the general nature of the downloaded game logs, then an investigation into the ease of extraction and use of explicit knowledge from these logs. As will be seen from

the next sections below, this effectively corresponded to working towards the outline of an implementation of the strategic layer component of the learning element of an agent, using a somewhat-supervised level of feedback, with a rule-based representation scheme.

Explicit Knowledge Extraction for the Learning Element of an Agent

Finding explicit data (i.e. rules) within the downloaded game log content was thought would not only prove emphatically and explicitly that there is knowledge within the game logs usable for agent implementation, but also provide knowledge in a form that could be used for a reasoning agent. Thus, implementation of the *learning* element (see 'Agent Learning' in section 1.2.2) of the agent would be concentrated on, in the hopes of outlining a *performance* element if particularly successful. In the process it was hoped that the remaining aims of the research (see section 1.3.2) could be resolved.

Learning for a Strategic Level Agent Component

Agent reasoning with explicit knowledge in a computer game environment can be correlated with the strategic level of thinking in an agent (see 'A Three Tier Model' under section 1.2.2). For this reason and because of the low granularity of data in the original set of game logs chosen to investigate (see section 4.2.3), it was decided that the usefulness in learning for a strategic level of agent thinking would be concentrated on. A strategic layer in an agent demonstrates the agent has reasoning capabilities that separate it from purely reactive models. When the TFC logs were found to be too sparse for real agent learning, this plan was continued with using the QUAKE II demo logs, a decision that, even with the time constraints and previous work, in hindsight was perhaps not suitably considered.

Somewhat-Supervised Level of Feedback

As the issue of major importance for the research centred around the investigation rather than implementation, it was decided that a level of supervision (i.e. human intervention) in the learning process would be necessary. This would initially be in the preparation and presentation of data. A somewhat-supervised level of feedback was indicative of the hope that this level of data preparation and presentation would be minimal, in keeping with the idea of a possible agent implementation.

Rule-Based Representation Scheme

The decision to use a rule-based representation scheme was largely based on the original idea to find explicit knowledge and the usefulness of rules (see section 1.3.1). It was believed these rules would be obvious from the captured behaviours of the expert players.
The problem of using game logs as a source of learning is one of induction:

> Using a set of examples f, find a function h that approximates f
> (Russell and Norvig, 2003)

where h in this case represents the original hypothesis that within the examples f (included as behavioural data within the game logs) are a number of rules that represent this behaviour strategically.

More specifically, for each single rule that is wanted to be found, using a number of examples, learn a rule that approximates those examples. At the risk of oversimplification, basically the collection of the rules found using this process should constitute the function that represents the behaviour found within the logs.

4.2 Implementation

4.2.1 Choice of Game Logs

Initially, it was believed that the use of logs that were not completely comprehensive in their description of player behaviour and environment would provide adequate learning data for agent modelling, especially if the data was made up of the contents of a large number of logs. It was thought that the limited behavioural and environmental coverage in logs of this kind would slide between a dearth of player behavioural data and being swamped with irrelevant data from which extraction of useful behaviour would be difficult. However, a search for repositories of FPS game logs found only repositories of one extreme or the other. For instance, logs found in a Team Fortress Classic (TFC) repository had been parsed automatically on upload to be human readable, meaning the content of the game logs was largely static, declarative, statistical and sparse. In contrast, various 'demo' game log repositories contained virtually continuous information for every aspect of the game that had been recorded, having been designed for enthusiasts to be able to replay the video of the entire game at their leisure. It was decided an investigation of the properties of both sets of logs would be of benefit.

4.2.2 Investigation of the World and Behaviour Content and Data

The first aim of this research was to investigate the content of game logs in order to identify the environmental and behavioural levels of data they contain. For the TFC game log files, which had been parsed to be useful for enthusiasts of the game, this basically required a perusal of examples of the logs online. For the Quake II demo logs, it required a retrieval process using a downloading agent and a parsing process with an API, as well as finding documents regarding the demo log file format. It seemed from this initial perusal that the TFC logs did not provide any environmental information, and a less than comprehensive description of player behaviour. However, it was hoped that using a large number of them would still provide insights to the game that could be of use to a learning agent at a strategic level. For the demo logs, it was found that the environmental and behavioural data was divided into two separate files. A video playback of a recorded game required both files. Investigation of the world content included looking at how the the world was modelled within the game environment, including the physics of moving entities and the in-file, three-dimensional representation and arrangement of static entities such as walls. Investigation of the behavioural content of the game logs meant playing, viewing and recording games, viewing the recorded outcome and looking at the form this behaviour was recorded in the actual game log files.

4.2.3 Investigation of the Extractability of Data from TFC Game Logs

While there was some apprehension over the granularity of behavioural data included within the TFC logs, it was believed they would make an adequate starting point. In addition, the source code for Half-Life was readily available, the repository did not include other FPS games, and being of a text-based nature, the data of the logs themselves would be easily extracted. It was still hoped that from a large collection of these game logs, in investigating the ease of extraction of data, behavioural trends could be found that would outline or at least point towards an actual agent implementation. As has been mentioned in section 4.1.2, investigation of the ease of data extraction centred around an implementation of rule-based data mining techniques in a bid to find examples of explicit knowledge for the use in the strategic component of the learning element for a reasoning agent.

Game Log Retrieval

As it was believed that the sparsity of the TFC logs could be overcome by mining a large number of them, an automated process was required to download a sizeable collection. As they were originally encoded as HTML pages, code was required to first download then parse the data in these pages into database tables where they could be used to build flat files for data-mining purposes. This process is described in (Wender, 2007). As it was expected a major issue with the demo logs would be a surplus of data, this process was not repeated with the demo logs.

TFC Data Preparation

Although originally over 10,000 logs were downloaded, not all were usable. After removing error-ridden, overly short and uneven team (due to bias concerns) game logs, around 6,000 were left over. Of the remaining logs it was found that over 2000 of them did not specify which map had been used, while the rest were distributed across 162 separate maps, with the most popular maps only having between 130 and 200 games each (Wender, 2007). Because many of the relationships that could be found given the type of data acquired from these logs were map specific (e.g. correlating team composition to game wins depends largely on the terrain as a sniper class player could score many kills on a map with good hiding places and expanses of open ground but do very poorly in a flat map with convoluted pathways) it meant the dataset would was effectively broken down even further.

Preliminary Analysis of the TFC Logs

In a preliminary analysis of the TFC logs, Wender (2007) attempted to use classification data mining methods to find behavioural patterns. Difficulties were encountered, firstly in the usability of much of the data and secondly in being able to find meaningful patterns. Two types of classification were attempted: the relation between the character composition of a team and the outcome of a game, and the performance of different classes and the outcome of the game. Neither returned conclusive results (Wender, 2007).

Secondary Analysis

A second attempt to find useful rules from the TFC logs was made by choosing useful attributes and placing them into a flat file, rules were searched for using classification and association rule learners (see chapter 2) using the Waikato Environment for Knowledge Analysis (WEKA) (Witten and Frank, 2005) and Brute rule learner (Segal and Etzioni, 1997). WEKA automatically breaks data sets into training and testing sets and provides a ten-fold cross validation tool to reduce the chances of overfitting. Though the Brute package provides a means to break the data set into training and test sets, rules found using it must be applied to a chi-squared testing procedure to ensure their validity (see section 2.3). While it had been planned to have a supervised component to the data preparation, the level required for the process did not seem to be reproducible in an automated learning setting.

Change of Direction

With analysis, it became obvious that greatest among issues, the logs were too sparse to be acceptable as a means of implementing an agent, even if that agent was given basic functionality to begin with (see section 5.2.1). While not useful for agent implementation, the results hinted at the difficulties involved in using game logs for agent learning. A change of direction was required, to logs with a higher granularity of data behaviour.

4.2.4 Investigation of the Extractability of Data from Demo Game Logs

The next attempt focussed on the use of downloaded demo game log files, of which a major problem lies in the abundance of raw data (see section 6.1.2).

Data Preparation of Demo Logs

The difficulty in finding and organising this data to prepare it for data mining was recognised and the original hypothesis was weakened again to allow a supervisory aspect over this part of the data mining process. It was still thought that mining for rules would be an effective way of learning from the game logs because of the large amounts of data that needed to be sifted through, the inherent rule-based nature of games and the several successful rule-based implementations of agents. The process

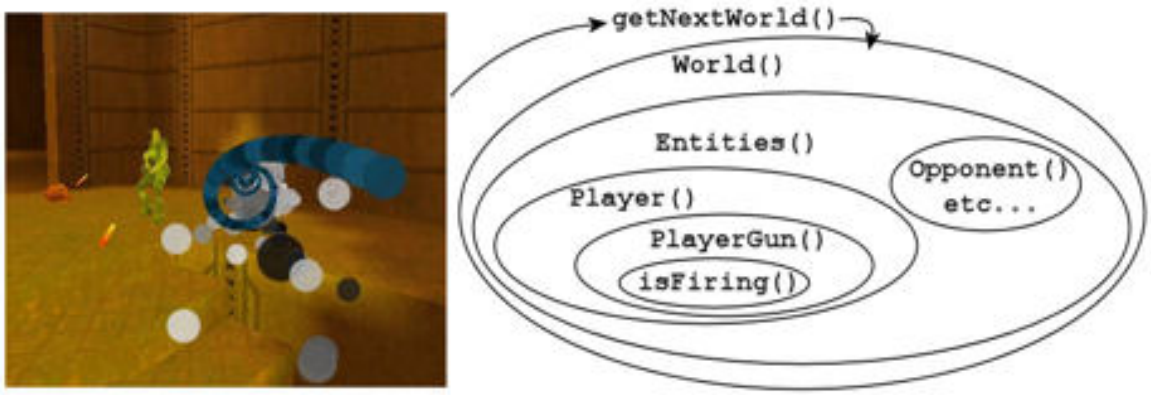

Figure 4.1: QUAKE II Demo Log Hierarchical Object Organisation: To find out if in this time frame the player was shooting, use the boolean isFiring() function, subsumed as part of the PlayerGun() object

amounted to extracting the values of selected attributes from the world state at each time frame directly from a demo log with the aid of the QASE API's getNextWorld() function, and saving these values in flat files, for use with classification and associative rule learners to extract rules from this data. Each time frame's world state includes hierarchical structures which organise attributes in an object oriented fashion. The value of a required attribute can be found by extracting the value from the object that attribute has been subsumed into, which can be found in a similar fashion, all of which is subsumed by the overall 'World' object (see figure 4.1).

Declarative Rule Mining of Demo Logs

The first attempt at finding rules in the downloaded demo logs was to try and find explicit rules in the form of what the human behaviour is attempting to accomplish. A supervised procedure segmenting quantitative data into qualitative attributes similar to the procedure used in (Lattner et al., 2005) was attempted to first represent the game log data in abstract terms. This required a large amount of effort for instance changing quantitative distances to qualitative values such as *very close*, *close*, *far*, and *very far*, to produce results that were not particularly satisfactory. Even with abstract representations, finding rules by subsuming continuous sequences behaviour to model a non-linear function was found to be very difficult. This was made even more difficult when these functions were further masked by human and/or game inconsistencies.

In addition, issues with causality arose, in particular with association rules, which are found by passing a certain coverage percentage, which can be as low as 5 or 10 percent (see section 2.2). It was not known how an agent would choose the correct association rule at the correct moment in time, if an array of rules could be found. By contrast, classification rules having by merit of their nature separated every attribute were thought to be able to potentially use rule percentages for each behaviour at each decision making instant to make a choice of rule.

Background Knowledge Component Investigation

The ever increasing supervised component of the implementation was felt to have reached a point where any rule learning done in an unsupervised manner would be unlikely. To combat decisions being made on which attributes were suitable, a side investigation of the possibility of creating a base of background knowledge for the agent was made. In QUAKE II, it was thought that the rules given with the game README file could provide a valid source for this background knowledge. Table 4.1 is a list of rules that could be relevant in this regard. It was thought that these manual start-up rules provided quite a good base for prior knowledge of QUAKE II. However, it was realised there were a number of implicit rules not covered by the explicitly stated rules that players are assumed to know, such as *a player kills someone by shooting them*, *the player will lose health if shot*, and *player will lose a lot of health if shot from close range*. A list of assumed rules is shown in Table 4.2. It was realised that both sets of rules are generally declarative in nature. This was in contrast with the inherent action-descriptive or procedural nature of the game logs themselves. With declarative game log data it was feasible to imagine that a number of rules could be learnt from the ground up (e.g. which gun to use when presented with an array of different weapon types), using deductive analysis and inductive logic programming processes. However, the procedural and quantitative nature of the data is at odds with this plan.

Procedural Rules

The third approach attempted was to search for procedural rules. These include the mistakes of humans, thereby creating a set of rules to imitate the player. Basic procedural rules considered are shown in table 4.3 which are the basic procedural rules programmed into Quakebot (Laird and van Lent, 1999; Laird, 2001). This approach's benefits include using the data as it has been presented and recorded, theoretically

Goal is to kill opponents more than they kill you
Each kill = 1 frag
Kill yourself, lose a frag
Throughout the map you find artifacts:
    weapons
    ammo
    health
    powerups
...that are needed to be successful at destroying your enemy
Weapons:
    Blaster: does not require ammunition
    Shotgun: uses shells for ammo, effective close range
    Super-Shotgun: slow firing rate, uses shells
    Machine Gun: aim gets lifted
    Chain Gun: good for sustained attacks,
        long spin up and spin down time
    Hand Grenade: Longer grenade held, farther you throw it
    Rocket Launcher: Be careful not to use in close quarters
    HyperBlaster: Chain gun with no spin up/down time,
        uses your energy cells Railgun: Powerful
    BFG: Very powerful
Armor: if you take enough hits, your armor depletes down to nothing
Health Packs provide an additional boost in your health

Table 4.1: Start-up Manual Quake II Agent-Relevant Rules

0 health is when you die
Kill opponents by shooting them
    Point gun at enemy and fire to shoot...
Lose health if you are shot
Lose a lot of health if shot from close range
Armor protects against health loss
Avoid being hit by moving and hiding
Direct hits are better
Closer implies better chance of direct hit
Person shooting at you is your enemy
Fall too far you hurt yourself

Table 4.2: Assumed Quake II Agent-Relevant Rules

Collecting Items:
Get most powerful weapons from item spawn locations
If item missing, remember when to come back
Get items using shortest path
If low on health or armor, get some
If a weapon is close, pick it up
Fighting:
use a circle-strafe technique when fighting
Depending on the weapon, move to correct distance from opponent
Chasing:
Use sound of opponent running to chase
If no sound, move to area the opponent was seen last
Ambushing:
Wait around corners where opponent cannot see you
If opponent is calculated to come out of a room, wait and ambush
Hunting:
If you kill your opponent, go to nearest player spawn location
Hunt in areas that opponent has been seen before
Collecting Items:
If not powerful item, collect if opponent not close

Table 4.3: Main Quakebot Tactics (Laird, 2006)

leading to a rule base (and from there, leading to behaviours) that directly mimic a person in the same situation. Perhaps this can be summarised as finding rules that replicate how to play a game as a person would. This approach is more consistent with the procedural nature of the logs, though is likely to produce only reactive behaviour. The most difficult aspect of this approach was found to be dealing with the human inconsistencies and inaccuracies and encoding these inconsistent behaviours as rules. Either the rules became dependent on probabilities or became heavily specific, unwieldy and incoherent, effectively destroying the most appealing aspect of rules - their accessibility.

The continuous nature of the data meant that if a qualitative discretisation method had been successful, temporal rule finding techniques (see section 2.2.1) would have been attempted. While investigated, it was realised that there were a number of issues that even temporal association rule learners were not going to solve. With a strategic level rule learning concept unusable, one final attempt at using static rule learners for tactical aiming rules was attempted.

Aiming Rules using Classification Methods

It was thought the best means for learning tactical aiming rules was to use a classification rule learner to have rules for every aiming possibility. Creating aiming rules requires agent relative data (i.e. opponent positions relative to the agent instead of a global positioning system) or rules numbers quickly explode with each only applicable in certain situations. Because the demo game log data contained only global positioning for entities, this required a supervisory process to translate global positions to agent relative positions.

Two methods were investigated, the first trying to minimise the translation process using gun angle and opponent distance and the second transforming the 3D global data to 2D positions (perspective projection). To simplify the process, only bullet-based weapon data was used, as in QUAKE II these weapons are modelled with instantaeneous fire (bullet hits target as soon as it is shot). In ideal circumstances, where the data being mined was qualitative and declarative in nature, an all-encompassing, simplified rule like 'point your gun at the opponent and fire to shoot' would simply and effectively cover all instantaeneous aiming conditions (rocket launchers, whose projectiles are slower moving require the player to shoot ahead of a moving opponent). Because of the procedural nature of the data, finding a rule like this was not possible. The first

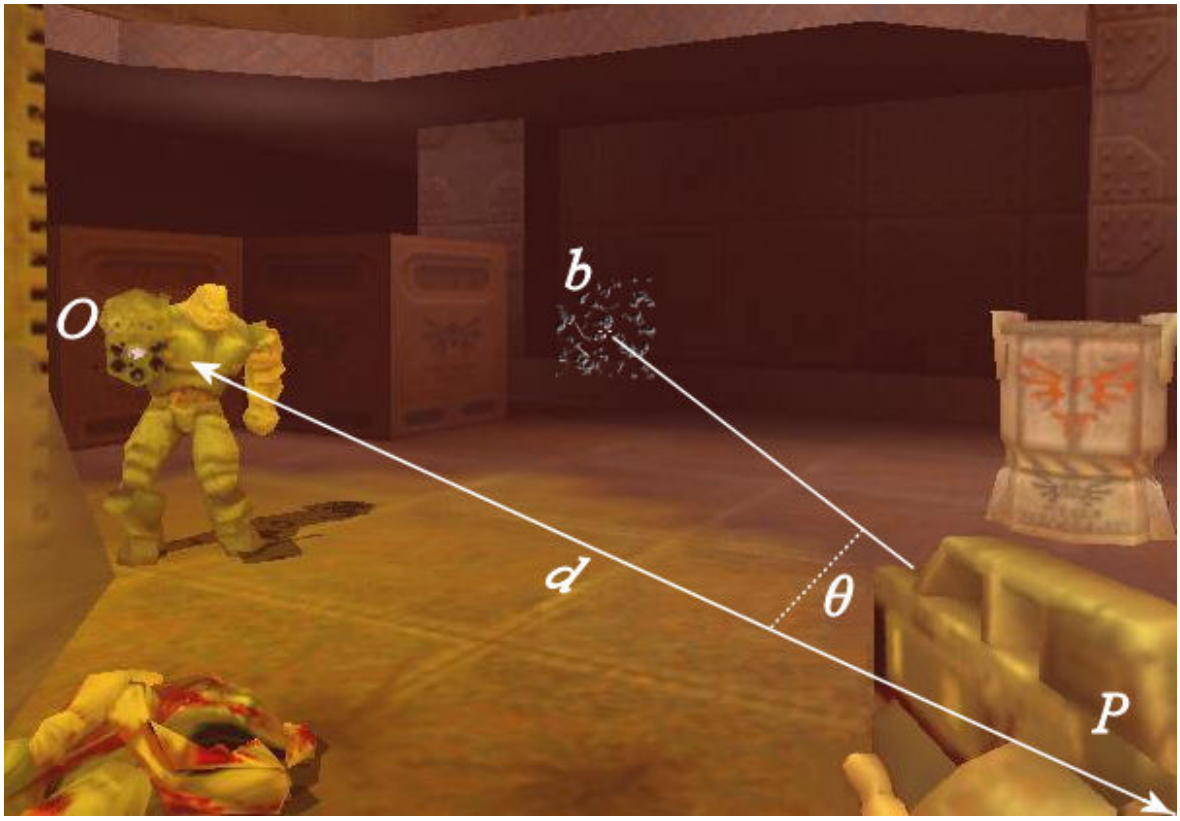

Figure 4.2: Simplified aiming rule discovery process: Comparison of distance d between player (P) and opponent (O) and angle θ between last shot fired b, player and opponent

method tried to simplify the aiming process and minimise the level of supervision by comparing the distance from the opponent to the angle between the opponent and the last shot fired (see figure 4.2).

Distance/Angle Classification Rule Method

Originally an attempt to manually discretise the data into height(*above*, *below*, *equal*), position(*left*, *right*, *infront*) and distance(*close*, *far*, *veryfar*) was made, but it was found that disparate instances can be found in three-dimensional positioning by using the same combination of these discretised values. A sub symbolic method was persevered with, with a goal of separating pitch and yaw rules. For the same reasons the discretised values did not work, this was not possible either. A single, unsigned angle was used instead, the process using trigonometric and vector calculations:

1. find game time instances where player is shooting

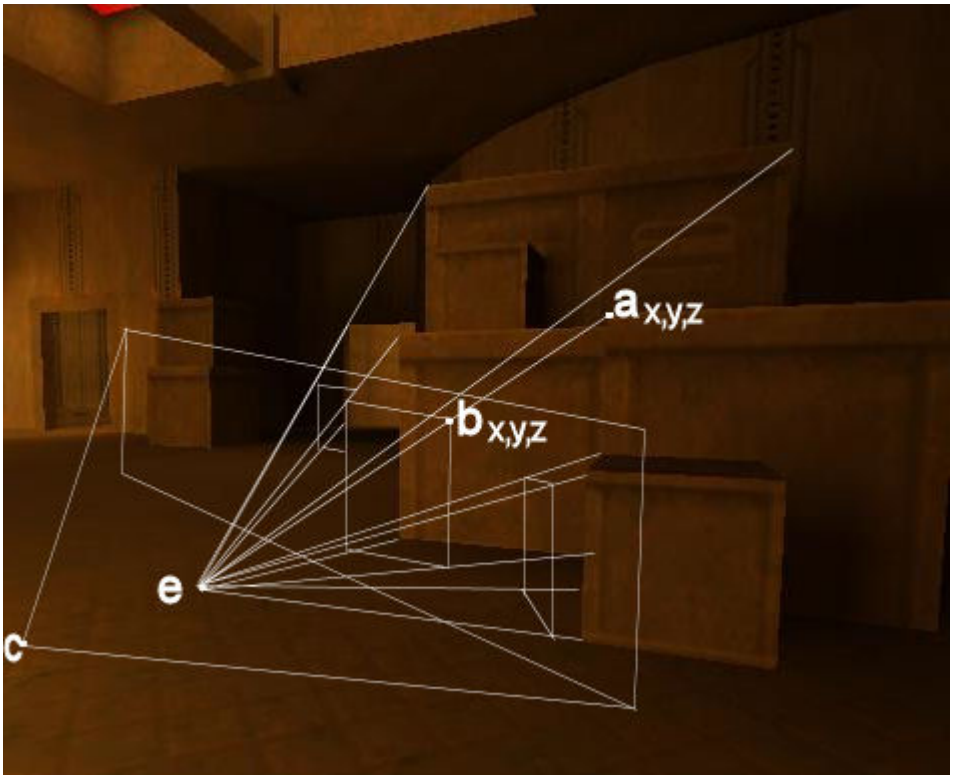

Figure 4.3: Example of 3D to 2D Perspective Change: a is the 3D point to be changed, b is its 2D equivalent, c is where the camera is positioned and e is the viewer

2. calculate distance between player and opponent using their global positions (x,y,z)
3. calculate and normalise the vector between player and opponent
4. calculate and normalise the vector between player and last shot fired
5. calculate the angle between these two normalised vector
6. for each time instance, note whether the shot was successful or not (hit the opponent)
7. classify hit/no hit using angle and distance

Perspective Projection Classification Rule Method

Perspective projection is a means to translate three-dimensional points to a two dimensional plane. An example of how a 3-dimensional environment is changed to a 2-dimensional perspective can be seen in Figure 4.3. Gorman and Humphrys (2006)

used a method similar to this is modelling a single tier tactical agent. Formulas for the method can be found in Appendix A.

Chapter 5

Results

This chapter presents the results of the investigation of downloaded, expert-level game logs as a source for general agent learning. It is believed that interactive computer game logs have the *potential* for use in concert with, or even replacement for, time-consuming supervisory learning processes for embodied, situated agents, but this potential is tempered by a number of issues that need to be overcome first. The data encapsulated within the logs encode the level of behaviour that has been sought by AI agent researchers in showing complex, multi-tiered behaviour as well as capturing nuances that when viewed can be recognised as being intuitively human. However, the inherent nature of the data in the logs themselves provides difficulties in the discovery and extraction of this behaviour. An unsuccessful attempt to use largely rule-based data mining processes to learn behaviours from game logs led to finding the inherently top-down nature of such processes was fundamentally at odds with the unsupervised bottom-up learning requirement of the problem.

5.1 The General Usefulness of Computer Game Logs

5.1.1 Content of Game Logs

Interactive computer game logs range from the purely statistical to others that capture the network traffic between the game client and game server several times a second, which integrated with the correct map and parsing tools can provide a researcher every action a player has made over the entirety of a game.

5.1.2 World Content

It was found that the parsed, TFC logs contained no environmental data at all. Files that corresponded to the world each log had been played on (i.e. map files) could be found, however, all links between the TFC logs and their map files had been severed in the parsing process. This is not an issue for players as they are more concerned with the statistics of their behaviour than of the map they were playing on. While the demo game logs had no environmental content either, they could be used in conjunct with the correct map file using a virtual server and middleware API to link player behaviour with world content. This meant that demo game logs, in concert with the correct map file, encode all aspects of a three-dimensional, complex, FPS environment and the entities within it that can include a number of mediums (land, water, lava etc), levels, powerups and teleportation devices. By storing the non-static entities and static environmental entities separately, the entire game information can be stored efficiently. The map files themselves describe the world using a binary space partitioning tree, with each leaf of the tree representing a region of the map.

Entity movements and positioning of static entities such as walls and stairs are linked to visual components of the environment that include transparent water, breakable and markable parts of the environment (i.e. gun shots leave marks on walls), and other environmental effects. While this research was not interested in the visual aspect of encoded data, their interactivity and realistic physics are interesting to note. Since QUAKE II is over ten years old, interactivity and visual realism can only have improved in the meantime, providing researchers with even more realistic environments to test agents within.

Several aspects of physics were noted to have been changed to improve gameplay, most notably the physics of ammunition. To provide a greater difference between weapons, bullet-type ammunition was found to have instantaneous firing properties. Rocket-type weapons, in contrast, were slowed to allow an opponent to see it coming and have time to dodge.

5.1.3 Behavioural Content

A TFC parsed game log would include several pages of statistics starting with an overall summary page including general features such as the number of kills, suicides and flag captures each player has been involved in. Other pages could be team, class of player

```
typedef struct    (
                      unsigned long size;
                      unsigned char messages[size];
                  )   block_t;
```

Table 5.1: Message Block Structure described as a C struct for simplification (Girlich, 2000)

```
typedef struct    (
                      unsigned char ID;
                      char messagecontents[???];
                  )   message_t;
```

Table 5.2: Message Structure described as a C struct for simplicity (Girlich, 2000)

(i.e. medic or soldier type) kills, flag activity or weapon specific. Statistics included on each page can be as specific as what time each player touched a flag, or the number of kills each player made with a separate weapon. The information is generally declarative, static (i.e. generally totalled numbers for the entire game) and sparse (players do not want to study every tiny aspect of the game, just get an overall idea of their play and general comparison to other players). An example of such a log is in Figure 5.1, the first summary page of a log from 'The Blarghalyzer' (Blargh, 2005) website. In comparison, a demo game log contains the positions of all entities within a certain radius around the recorded player every tenth of a second. These demo logs are unreadable with a normal text editor (see Figure 5.2). As mentioned demo logs contain the sever-client network traffic every tenth of a second. The file stucture is very complicated, consisting of a set of message block structures which contain a 4 byte length entry and the set of messages for that block(see Table 5.1). Each message consists of an ID string and the actual message (see Table 5.2). A message can be representative of a temporary entity (such as a bullet), a player's inventory or statistics, information about a certain time frame or server/client operations (such as *nop*, or *disconnect*). See (Girlich, 2000) for a full listing.

To be able to parse these QUAKE II demo and map files, an application programming interface (API) is very useful. One of the reasons the QUAKE II logs were chosen was due to the availability of such an API, called the QUAKE II Agent Simulation Environment (QASE) (Gorman et al., 2005).

The player behaviour found in the demo logs was expert-level: players were competent in all areas of gameplay, using powerups and the surrounding environment to their

The Blarghalyzer - TFC Game Server Log Parser

Summary | Blue Classes | Red Classes | Blue Buildables | Red Buildables | Flag Activity | Weapons | Public Chat | Blue Chat | Red Chat | All Chat | Download Logs

Server: FBD UGC 7on7 Hosted by .AI
Map: high_flag Start: 06/01/2008 20:30:39
Blue Score: 80 End: 06/01/2008 21:00:39
Red Score: 60

Blue Team:

Player	Classes	Kills (FC)	Deaths (FC)	Suicides	Team Kills	Touches (I)	Caps	Flag Time	SGs Killed (L1, L2, L3)	Game Time
|FBD|wAnnA fAntA?!?	Snipe / HWG	101 (12)	27 (1)	3	2	1 (0)	0	0:00:00	0 (0, 0, 0)	0:30:00
<(-=*[=Dramling=]*=-)>	Engy	71 (10)	29 (1)	4	5	1 (0)	0	0:00:02	0 (0, 0, 0)	0:30:00
<(-=*[=carrrrrrrmneee=]*=-)>	Engy	51 (6)	22 (0)	2	0	0 (0)	0	0:00:00	0 (0, 0, 0)	0:30:00
HotKarl	HWG / Scout / Med	41 (1)	79 (9)	5	1	12 (5)	3	0:00:46	16 (0, 3, 13)	0:30:00
Dan.AI	Scout / Med / Sol / Pyro / Spy / HWG	34 (4)	42 (5)	5	0	8 (3)	2	0:00:41	2 (1, 0, 1)	0:30:00
StarScream	Med / Sol / Scout	32 (0)	68 (8)	1	2	10 (0)	0	0:00:09	7 (1, 2, 4)	0:30:00
ODiE	Scout / Med / Sol / Spy / Demo / Pyro	24 (0)	82 (8)	6	2	13 (0)	3	0:00:49	2 (1, 0, 1)	0:30:00
----------TOTALS----------		**354 (33)**	**349 (32)**	**26**	**12**	**45 (8)**	**8**	**0:02:27**	**27 (3, 5, 19)**	**3:30:00**

Red Team:

Player	Classes	Kills (FC)	Deaths (FC)	Suicides	Team Kills	Touches (I)	Caps	Flag Time	SGs Killed (L1, L2, L3)	Game Time
[SNAFU]sponge	HWG	111 (10)	28 (0)	2	3	0 (0)	0	0:00:00	0 (0, 0, 0)	0:30:00
[SNAFU]Hellfish	Engy	75 (10)	44 (0)	11	4	0 (0)	0	0:00:00	0 (0, 0, 0)	0:30:00
[SANFU]volt	Engy	67 (9)	32 (0)	7	6	0 (0)	0	0:00:00	0 (0, 0, 0)	0:30:00
superegg	Sol / Med / Demo / Pyro / Spy	51 (1)	71 (3)	6	7	5 (0)	0	0:00:16	19 (2, 7, 10)	0:30:00
{GCCr} M C o Z	Spy / Sol / Demo / Pyro	22 (2)	15 (0)	3	1	0 (0)	0	0:00:00	0 (0, 0, 0)	0:30:00
[SNAFU]liquid-//-|BSO|	Scout / Med	13 (0)	70 (13)	5	0	17 (5)	1	0:00:44	3 (0, 1, 2)	0:30:00
snafuprimordialfollicle	Scout / Med / Snipe	10 (0)	94 (17)	2	2	24 (6)	5	0:01:05	4 (1, 2, 1)	0:30:00
----------TOTALS----------		**349 (32)**	**354 (33)**	**36**	**23**	**46 (11)**	**6**	**0:02:05**	**26 (3, 10, 13)**	**3:30:00**

FC = Flag Carrier I = Initial Flag Touch

Events Summary:

Spawns	Conc Jumps	Caltrops	Discovered Spies	Detpacks	Infections	Heals	Infections Healed	Disarmed Detpacks	Tranqs	Gassings	Conced Kills
390	194	0	0	0	0	15	0	0	0	4	26

Spawns	Conc Jumps	Caltrops	Discovered Spies	Detpacks	Infections	Heals	Infections Healed	Disarmed Detpacks	Tranqs	Gassings	Conced Kills
415	272	0	1	0	1	2	0	0	0	5	47

Figure 5.1: Summary page of the parsed Blarghalyzer Logs

Figure 5.2: Gamelog viewed in a text editor

advantage. Players were only still when waiting to ambush an opponent. Complicated manoeuvres made up of smaller basic moves were commonplace. A good example of this is the 'rocket jump', an expert-level move to reach otherwise unreachable areas:

1. While running forward, the player aims the rocket-launcher at his/her own feet
2. The player jumps and with careful timing shoots a rocket into the ground below
3. With practice (and some luck) the blast energy from the explosion of the fired rocket below propels the player higher than possible from a normal jump.
4. The blast energy still has some effect on the player's health. Wrongly timed or if too weak when tried, the sequence can be fatal to the player.

In capturing data at every tenth of a second, uniquely human characteristics or ticks are also captured. These include hesitation (e.g. before stepping out into the open), changing a plan or task mid-way, or jumping or moving around unnecessarily. While perhaps not logical, for other people, these behaviours can be instantly recognisable and the intention behind them understood intuitively. This was taken advantage of by Thurau et al. (2005) to model a reactive tier of an agent (see section 3.3.2).

For opponent modelling, the capture of opponent positional data in demo game logs does not depend on a line of sight, just if the opponent is within a certain player radius. Because it is not necessary for rebuilding a video of the game, knowledge such as the opponent's inventory is not encoded.

While the data is comprehensive, it is by nature procedural, continuous and quantitative. Procedural as it describes the actions of entities, continuous (but not necessarily sequential) as one event follows directly from the next and quantitative as the data is stored as numerical values.

Finally, because the data in the logs is primarily to rebuild the game with a map file to be viewed as a video for enthusiasts, the majority of behavioural data is encoded as positional data, that is, where entities are positioned at each time instance. This

means that any behavioural knowledge encapsulated within these logs is implicit rather than explicit.

5.1.4 Game Logs and Convenience

By encoding the behaviours described above and being available in large quantities from a number of online repositories, these expert-level game logs have an inherent convenience for researchers interested in general agent modelling by providing 'ready-made' training examples (if behaviours can be extracted).

5.2 General Issues found with Game Logs for General Agent Modelling

The attempt at using the TFC game logs showed primarily there is a granularity of behavioural data required for agent learning. It also raised issues with the difficulty in data mining methods requiring the user to already know what they are looking for, though this was not recognised until later as this difficulty was laid primarily on the sparsity of data of the TFC logs, as it had been so difficult to tie together any two rules at all. The general issues presented here represent the order they were come across during this research: sparsity and then overabundance issues; the symbol grounding problem; procedural and continuous data problems; background knowledge; and finally agent relative data.

5.2.1 Sparsity/Overabundance of Data

With time it was obvious that the initial apprehension over the granularity of behavioural data in the TFC logs had been founded. It became obvious that the logs were too sparse to be acceptable as a means of implementing an agent, even if that agent had been given basic functionality to begin with. Rules that could have been found had little relevance to each other (rules too sporadic for each character class to build a complete agent), were spread over nine different character classes (e.g. medic character class infection rates for different characters or the number of sentry guns built on average by engineers on each map) and at best gave a broad observation as to how that character was generally played by the gaming community (i.e. most popular weapon of one class to kill another class of character).

5.2.2 The Symbol Grounding Problem

The symbol grounding problem, formally introduced by Harnad (1990), refers to the problem of how symbols are linked to the things they denote. Formal or symbolic systems can be seen as collections of axioms, which are created as finite sequences of abstract symbols (statements or sets of symbolic rules). Unless there is a means to interpret these symbols, to an outsider looking into the formal/symbolic system they are effectively meaningless. This problem was taken advantage of by Searle in his famous Chinese room argument against strong AI (Harnad, 1990). In a FPS environment, an interpretation of a formal system requires 'grounding' of the abstract symbols by linking to the sub-symbolic (quantitative) data each symbol represents. Given the kind of rules that were hoped to be found would point to integration in a declarative agent, this was a significant problem.

5.2.3 Continuous Nature of Data

If a player's actions, framed as continuous data in demo game logs, are to be broken down into a set of rules or behavioural modules or some form of state/transition plan, it can be seen that a number of actions are more important than others - thereby necessarily interrupting the actions of other less important ones. This means behavioural modeling of agents in FPS environments, as in real life, whether taking a symbolic or sub-symbolic approach, must have hierarchical properties with some actions being necessarily interruptible. In symbolic terms, this means any discovered rule or behaviour must be subsumed into the correct level of some form of hierarchical behavioural model. This problem as has been mentioned has been investigated by Nejati et al. (2006), and has been applied to modelling goal-defined exploratory traces in a FPS environment (the difficulty in creating relative, goal defined behavioural processes is addressed in the next two subsections). In sub-symbolic terms, it would mean the updating of the correct combination of behavioural modules. In both cases, as perhaps already evident, some form of cognitive architecture is probably necessary before learning begins.

To circumvent this issue, the supervisory level within the learning process was increased, by attempting to break down actions arbitrarily and then learning these smaller behaviours. However, it was found that in many cases the player would break off mid-task to pursue a more important action. For example, should an agent be walking towards an ammunition pack when an opponent appears, an action to protect

itself against attack should generally override the current action of collecting items to increase its overall efficacy in attack. Using rule-based data mining processes requires consistency of actions to find rules of high accuracy. This consistency could not be found using the expert logs downloaded from the internet. The next level of supervisory control would be to create game logs with behaviour that was ideal. It was felt that this level of supervision compromised the original hypothesis too far: an inherent potential usefulness in game logs is their ready-made nature, so creating basic logs to find data to imply the usefulness of expert level logs did not seem reasonable.

5.2.4 Background Knowledge

The attempt to include a base of background knowledge to circumvent the mentioned issue of agent learning from game logs requiring a low supervisory setting was unsuccessful. This was largely due to issues with the procedural nature of the logs themselves.

5.2.5 Procedural Nature of Game Logs

As it provides quantitative data of the actions of the player at every tenth of a second time step, the QUAKE II game log data was found to be mostly procedural in nature, that is, describing how behaviours are performed, rather than framing behaviour in intention/state annotative actions. This is in contrast to the TFC logs, where it is believed the Blarghalyzer's automatic parsing process was probably the cause of their static, declarative nature. An agent using only procedural rules is imitative and reactive in nature, which was an approach that was initially shied from.

The procedural nature of the data means the path taken by the player through the game is the only path to learn from. Even with other logs and examples, decisions for specific instances are limited by this fact. Though it might be possible to work out the intention behind every action given the overarching game rules and game objects, building declarative intention-action behaviour from these logs is thought to be a difficult problem.

It was found that procedural rules suffer the same general issues as declarative ones. In order to try and force declarative, then procedural rules to work with the background knowledge rules out of the QUAKE II demo log data, higher and higher levels of supervisory involvement was assumed in the learning process to the point that the original hypothesis had been compromised and the potential strengths of game logs had been subverted. This is further discussed in section 5.3.

5.2.6 FPS Game Log Data not Agent-Relative

Agent-relativity of data implies the data captured is from the agent's point of view. This is important quality of data for agents to be able to learn a smaller number of generalised and understandable rules. Current interactive game log attributes are recorded in a global positioning system, that if used, leads to an explosion of highly situational (global) specific rules. Agent-relative data could be calculated from these attributes, but it required a significant amount of preprocessing. Utilising game log data for modelling aiming behaviour required the use of complex trigonometric processes to be applied to the data (perspective projection-see Appendix A for formulas) to implement agent relative shooting behaviour.

5.2.7 Game Log Data in an Unintuitive Format

Similarly, while the data in the demo files is extensive, much of it is difficult to subsume into rules because the architecture of the server/client is based around placing various objects as they appear in each instance where they should be in the game world or level. This means that for something as simple as aiming the only information we have is the position of the player and his opponent, and the way the player is aiming his gun. While we have a lot of data, the data is unintuitive; the relationship between the pitch and yaw of the aimed gun and the positions of each player is trigonometric. This can be seen in figure 5.3, which plots bullet shots and whether they have hit the player or not. To increase the number of instances, both opponent and player shots were considered. As can be seen, the positioning of the gun shots within a certain area does not absolutely entail that a person has been shot, as there are misses within the area that correspond to a certain area around the central position of the player.
The unintuitive format means two things:

1. It is likely that especially for symbolic approaches, game logs will not be useful until game developers see their potential for NPC intelligence and format them in a way that is more intuitive (see section 6.3.1).

2. Until this is the case, for any approach taken, symbolic or sub-symbolic, a large amount of pre-processing of data will be required.

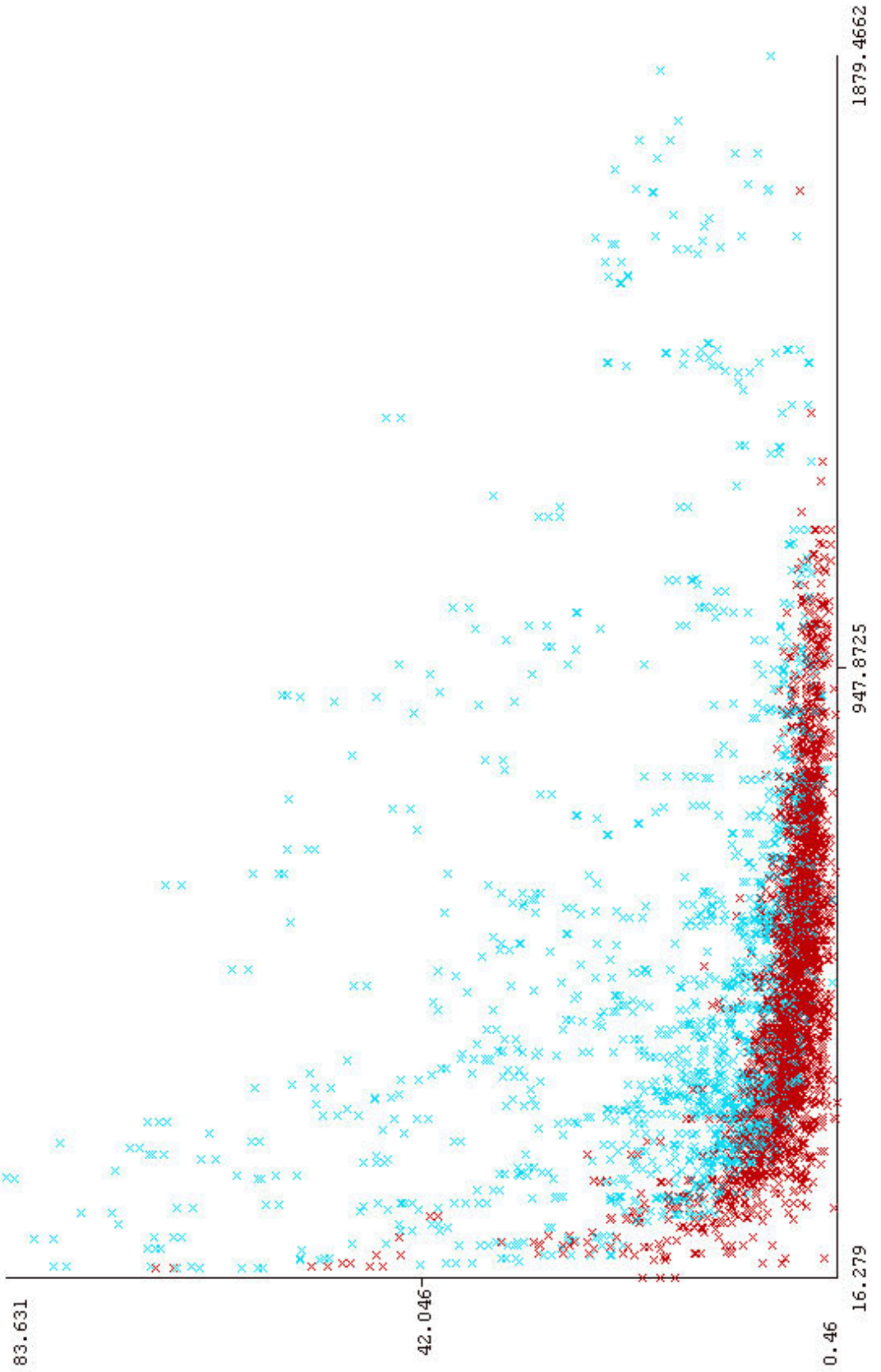

Figure 5.3: Results of 3D to 2D Perspective Change on Bullet Shots Fired. Red = Hits, Blue = Misses

5.3 The Use Of Rule Based Data Mining Methods with Game Logs

It has been found that rule-based data mining methods, somewhat unintuitively, are not of real use in the area of agent learning from First-Person Shooter (FPS) interactive computer game logs. It was believed the inherent rule-based structure of games, previously successful hand-coded, rule-based agent implementations and the extraction process necessary from interactive computer game logs would point to the method's use. However, it was found that the top-down and statistical nature of rule-based data mining is at direct odds with the bottom-up and causal requirements of agent implementation when using game logs. As mentioned, supervisory levels are needed to be kept low as it is easier to produce single logs and elucidate every action separately in (basic) movements for learning agents than it is to extract behaviours from the readily available, expert-level game logs by hand.

5.3.1 Issues with Rule-Based Data Mining Methods

Needing To Know What to Know

It would seem that using data mining techniques would be useful, especially in regard to the large amounts of data that are present and with current data mining techniques providing another means to bring quantitative, inherently time-based data like quantitative player behaviour to a symbolic level using knowledge based temporal abstraction (Shahar, 1997). Unfortunately, as will be discussed further in section 6.1.2 rule learning from even qualitative data temporal sequences seems to still require top-down methods of knowing the patterns and relationships you are looking for before being able to find the sort of relationships between each sort of pattern (Sacchi et al., 2007). Experiments in using data mining techniques to find rules in game log data emphasise the inherent fundamental discord with the view of using game logs as a means of expert behavioural traces instead of laborious supervised techniques and a golden rule of data mining being knowing what you are looking for (Pyle, 1999). Even when using brute force techniques, rules found must be recognised as being valid.

Statistical Rules and Causality

Rules found by inductive methods are use statistical notions of 'coverage' (see section 2.2.2) to select appropriate rules. However, a strong rule of statistics is *cum hoc ergo propter hoc* or 'Correlation does not imply causation'. This poses problems especially for association rules, as a decision making process require knowledge of which rule to fire when.

Low Level Nature of Data in Game Logs

To build a general understanding of a concept from base data is difficult if the low-level data is not a logical building block to that point. This is evident in the current research in that much, if not all of the behavioural knowledge is encoded in implicit terms.

Chapter 6

Discussion

This chapter discusses results found and then outlines possible approaches to the problem. It first discusses the general usefulness of game logs for agent modelling, that is, why they provide the potential to supplement or even replace parts of a supervisory agent learning process. Secondly, it discusses the general issues found with game logs for agent learning, that is, issues that span a variety of methods and philosophical approaches. From there, a discussion of the usefulness and issues specific to a rule-based approach, including discussion of the methodology of this research. Finally, there is a look at possible approaches that bridge the issues discussed in the chapter by subsuming successful applications of various methods in narrower fields. As the almost purely statistical TFC logs were found to be too sparse (see section 6.1.2), this chapter will mostly concentrate on demo logs, with particular reference to the QUAKE II demo logs used in this research.

6.1 On The General Usefulness of Downloaded, Expert-Level Computer Game Logs

The use of game logs for agent modelling is seen here primarily as a near limitless source of expert behaviour for unsupervised agent learning, potentially limiting or removing the need for researchers to create their own examples for agents to learn from. This section first discusses the incentives for using game logs for agent learning by looking at their environmental and behavioural content. Next it discusses the innate conveniences of game logs such as their facilitation of offline learning processes and their use as a means and source for automated data capture.

6.1.1 Content of Game Logs

The environmental and behavioural data that is captured within the downloaded demo log files and their corresponding map files is the main drawcard for using interactive computer game logs for agent learning purposes. This section first discusses the environmental content with respect to the realism, interactivity and potential for embodied, situated agent research, then the potential of the link between the visual and underlying data. From there a discussion of the behavioural content will be made, including its uniqueness, humaness, expert-level and motivated foundations.

Environmental Content

FPS environments are three-dimensional, complex environments that can include a number of mediums (land, water, lava etc), levels, powerups and teleportation devices. They are complex, realistic and require task and goal-oriented behaviour of agents in real-time. Such an environment is important for agent research because it is able to test an agent in many ways and on many levels of cognition (i.e. reactive, tactical and strategic). It is particularly suitable for general agents in that agents that have been programmed to specialise in one particular task very well are quickly found out and taken advantage of in real-time play (Laird and van Lent, 2000).

Besides the environment's complexity is its interactivity. Water can be fallen into, objects can be broken and walls can be marked with gunfire or explosions. This provides a rich sub-set of tasks and learning for agents as most actions have consequence and provide some form of feedback.

FPS avatars are human-like in shape. Avatar actions are made to resemble human actions (such as lifting an arm to press a door switch) so agent learning can incorporate ideas of embodiment and situatedness in a human form, without having to deal with the mechanical issues of creating a humanoid robot, in an environment where sensory data is not noisy. Laird and van Lent (2000) in particular notes the potential of computer games for the study of 'human-level' intelligence. While many of the physics of these FPS environments are changed for ease of programming or gameplay (e.g. slowed rockets and instantaeneous bullets) like AI (see Laird and van Lent (2000)) the realism of game physics is defining the next generation of games (Henry and Karsemeyer, 2008).

General FPS agents can be considered situated in these environments as they are surrounded by and are able to sense and interact with almost every aspect of it continuously. They can also be considered embodied as these senses and interactions with the

environment are based on the effects seen on and with the avatar they are instantiated as. Adding the breadth of tasks required to survive in these FPS environments to the principles of situatedness and embodiment, it can be seen that agents modelled using these environments are attempts by researchers at modelling complete agents (Pfeifer and Scheier, 2001).

The map files themselves encode the positioning of objects around environment as a binary space partitioning tree. This data can be accessed in conjunction with behavioural data to work out surroundings for any time frame instance, meaning the settings around which the behaviour played out is as accessible as the behaviour data itself.

Because this data is linked by time frame instance to the visual output of the player graphical user interface, these logs provide hours of footage describing as well as picturing a scene. Among other uses, this property of the logs could have relevance to problems faced by researchers dealing with agent vision systems or those in the graphics community interested in bridging the semantic gap in multimedia (e.g. see IBM Research (2008)).

Behavioural Content

The behaviour captured within the downloaded demo logs is unique for being spread over so many different players at an expert level, a variety and expertise that cannot be matched by a researcher creating behaviour from scratch. The behaviour is also unique for its fundamental motivation; players are having fun, want to be doing the actions that are captured and at the same time want to win. It could be that people being so immersed in the fun of their gaming experience (as a means to escape the real world for a time) have greater concentration at the task at hand than captured real-life behaviour might. Occasions when people 'really let their hair down' are rare and these games could be a source for this kind of behaviour as well.
Behaviours captured include aspects of reactive, tactical and strategic decision making:

- Reactive: spinning to shoot an opponent who has ambushed a player or making a decision to run
- Tactical: aiming ahead of a moving opponent with a slow projectile weapon in order to hit him
- Strategic: using knowledge of the game to enhance overall gameplay

Implicit Intention FPS are renown for the fast pace of action, stressful situations and quick reactions needed to survive (Schreiner, 2002). However the simplicity of the game play can be seen from a researcher's point of view one of the strengths of the behavioural content. While intentions at specific moments might be difficult to discern, overall intention (i.e. to win the game) is always obvious, meaning it could be possible to work out intentions for each sequence of actions a player makes.

Gameplay The simplicity of the gameplay does not mean play cannot be nuanced: though in our minds FPSs are attacking games where one goes out to kill your opponent a greater number of times than they are able, by the time it reaches an expert level the game becomes defensive oriented. That is, the aim is to die less times than your opponent does. This means attacks become more subversive, tending toward brief lashings out with powerful weapons followed by flight. Knowledge such as the positions of health packs and armour becomes paramount to a winning strategy.

Believability of Agents Current agents in computer games suffer from not being believable and human. For instance, Sengers notes:

> "The divide-and-conquer methodologies currently used to design artificial agents result in fragmented, depersonalized behavior, which mimics the fragmentation and depersonalization of schizophrenia in institutional psychiatry. The fundamental problem for both schizophrenic patients and artificial agents is that observers have difficulty understanding them narratively" Sengers (2002, pg. 96).

Gorman et al. (2006b) note that small human nuances can solve this issue (see section 6.3.2). These nuances include intuitively recognisable human ticks like hesitation, unnecessary jumping or decision changes. Though not logical, these small behaviours add to the viewer's belief of the agent's humaness by seeing actions that are unnecessary but *narratively* correct.

Opponent Modelling A final mention of the behaviour potential in downloaded demo computer game logs is the potential for modelling the behaviours of opponents, a key requirement in tactical play for games (Kuhlmann et al., 2006). The capture of opponent data in demo game logs follows an intuitive pattern: if the opponent is within a certain player radius, his behaviour is encoded just as the player's is. Within this

radius a real player might be able to discern footfalls, if not a direct line of sight. The timing of the disappearance of opponent data can be linked to the point where the player can no longer be certain of his opponent's whereabouts.

Convenience of Computer Game Logs

As has been mentioned in section 1.2.4, expert-level interactive game logs are readily available from a number of repositories on the internet. Downloads are free, and there is a large number available for each game. As an automated form of behavioural data collection, it is a system that seems hard to beat. The files can be seen as a source for offline learning for agents that takes no effort to create and a small effort to collect (see Wender (2007) for a automated downloading process).

6.1.2 General Issues

The general issues presented here represent the order they were come across during this research: sparsity and then overabundance issues; the symbol grounding problem; procedural and continuous data problems; background knowledge; and finally agent relative data. Possible solutions to many of these issues are presented in section 6.3.

Sparsity/Overabundance of Data

Sparsity and overabundance concerns were respective dichotomous features of the downloaded TFC and demo game logs. These concerns straddle the fine balance between inconsistency and granularity in data representation, as Poole et al have noted:

> "The richer the representative, the more useful it is for subsequent problem solving. The richer the representative, the more difficult it is to learn" Poole et al. (1998).

The use of the TFC game logs showed that there is a granularity of behavioural data required for agent learning. For learning that includes agent positioning around the complex environment, it could be that the level of granularity seen in the demo logs is necessary. This sort of level also captures not only the nuances of human ticks as mentioned in section 6.1.1, but might be able to be used to also build sufficient models of intention, if a symbolic approach is taken.

While a sparsity of information means inconsistencies in inductive learning approaches, an overabundance of data means issues for unsupervised learning algorithms,

in breaking down the dimensionality of the data for each time instance, but also for choosing which attributes within the data are useful for any specific learned component of the agent. For explicit knowledge learning, an important single rule can be a small instance in the large amount of data. How this instance is separated from any number of other instances is a difficult problem.

The Symbol Grounding Problem

Due to the quantitative rather than qualitative nature of the data, for symbolic approaches the symbol grounding problem is a significant issue. Instances cannot be framed and behaviours cannot be classified declaratively without significant reorganisation and labelling of data. This would amount to a complete overhaul of the original data, or a large supervisory component within the learning, which as has been mentioned in section 5.2.3, is against the fundamental advantages of using game logs for agent modelling. For example, in attempting to find aiming rules, one time-frame's captured world state might contain the positional coordinates and angles of player and opponent but no labelled links to what these positions and angles represent (i.e. how coordinates of player's position is linked to the abstract concept of the player itself, without having prior or given knowledge). Konik and Laird (2004) circumvented this learning problem for their symbolic agent in an adventure game setting (where the decision-action process is linear in nature) by getting an expert (human) to elucidate every action. It was thought this might be applicable to the current research, by parsing the extracted parts of the data into declarative instances. Eventually, this was seen to be infeasible for the FPS setting we were using, which requires hierarchies of interruptible actions depending on the situation and current strategic goals (see Continuous Nature of Data subsection below).

Temporal data mining has discretisation methods to segment continuous data and algorithms for assigning abitrary labels to these discretised segments (see section 2.2.1). However, these methods need sequential data and means of assigning labels often rely on previous definition of segment shapes (Agrawal et al., 1995). Rules found are propositional unless data can be transferred as attributes to database tables that include relationships between them (see section 2.2.2).

Continuous Nature of Data

Deriving basic lessons by watching experts means the ability to break down and slow down complex manoeuvres into their constituent smaller manoeuvres, an exercise that effectively requires expert knowledge to begin with (see Background Knowledge paragraph below). The highest level behaviour shown by expert play is repeated often and can be quickly identified by people viewing demos. Especially when large damage weapons are being used, there is a set pattern of behaviour between the players: a brief encounter, one or two shots fired by each player, followed by dashing away, circling (often quite a distance away from the initial encounter) to another vantage point for another fleeting conflagration, etc until one of the players has been killed. While a repetitive behaviour and recognisable again and again to people, the behaviour is hard to pin down in game demo data: the instances are fleeting, the circling being of different distances and paths (especially if handy powerups are along the way) or not even circling really, with the enemy moving from his position too.

Effectively the players are following a task hierarchy, this high level task (shoot, run, attempt to ambush, repeat) can be broken down into several more basic tactics:

1. if opponent has a one-shot kill weapon, shoot and run (ideally put something solid between you and your opponent)
2. when attacking, use a circle strafe behaviour
3. move to the best distance for you current weapon
4. pickup close powerups (to foil opponent pickup as well as for own benefit)
5. use ambush tactics
6. attack enemy from behind if possible
7. wait by a position the enemy is expected to come out to

A simplified version of this can be viewed in Figure 6.1. This manoeuvre has implicit within it tactical weapon control (which weapon to use, when), modified circle/strafing behaviour, map knowledge and ambushing behaviour, with each constituent part seamlessly blended into the others parts - a manoeuvre which, due to the map specificity, is unlikely to be repeated exactly again. To learn this behaviour, an agent would not only have to learn these tasks separately, but learn in which place in a hierarchical

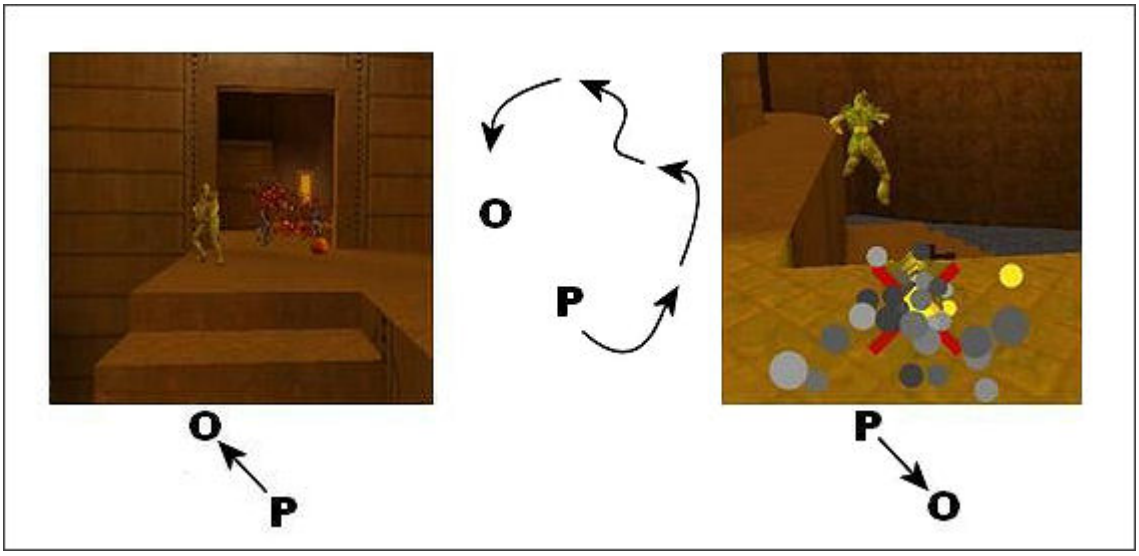

Figure 6.1: Simplified expert player (P) behaviour example (opponent (O) does not move): Reacting (firing), changing weapons, circling, ambushing the opponent

task network they fit. The tasks involved in the overall higher order tactic are melded seamlessly together. It would be very difficult to separate the tasks in order to work out which interrupt others or are of more importance.

The problem is especially difficult when taken from an observer's point of view. For instance, working out whether an expert player ran through a powerup specifically or incidentally (i.e. the powerup happened to be on the player's path). If the player ran through the powerup specifically, it means the player has diverged from his current path to take advantage of the nearness of the powerup that might be of use later. This means he has momentarily stepped out of one task to fulfil another and so could affect the overall task hierarchy. If the powerup pickup happened incidentally, the player is totally focussed on the task at hand but has run over a powerup accidently, meaning the task hierarchy should not be updated. Perhaps with extra checks such as if the powerup were a health pack and the player's health was low it could be assumed that the player had changed his task momentarily. Even so, such assumptions add another layer of complexity and add to the noisy nature of the data.

Background Knowledge

Tabula rasa or from scratch agent learning (bootstrapping) is a difficult process that requires a high granularity of training data and highly supervised or largely imitative

learning processes. It has been mentioned in the introduction (and discussed further in section 6.2.2) that in using game logs for agent learning it is inappropriate to use largely supervised methods. This leaves imitative processes, which although have had promising results in agent learning from game logs (Gorman et al., 2006b; Floyd et al., 2008), to date have not led to multi-hierarchical, task enabled agents. If an inductive process is to be persevered with, involving background or prior knowledge can be seen as adding an analytical (deductive) learning process to the overall agent (Mitchell, 1997).

Anecdotally, people need directions when they are completely unfamiliar with a subject. This can be in the form of a map, directions for putting furniture together or a README or help file on how to play a game. While it could be argued that a large number of people tend to disregard such information and dive straight into things (especially when the instructions are unintelligible), they do so with at least concepts of the basic principles of their endeavour. For instance, taking the previous examples of map reading, furniture construction and game playing: the person involved could know the general area of where they are going to and can rely on street names when they get close enough to their destination; he or she could have seen the shape of the furniture on the side of the box and deduce the positions of separate parts from their shape; or they could have played first person shooters before and can use this experience to immerse themselves immediately into their new purchase.

However, experience does point to situations when disregarding instructions leads to situations worse than those originally started in: getting lost; having screws left over and a decidedly unstable couch; getting stuck and thoroughly frustrated at a situation in a game that requires specific knowledge of new features explained clearly in the README file. It could be said these anecdotal situations are failures of a person's analytical learning process: the output hypothesis based on a deductive analysis of prior data did not adequately fit the current domain, and a reversion to more difficult and time consuming learning schemes are in order.

Still, such floundering in a situation is worse when no idea of the activity is known at all and rules are expected to be known simply from viewing that activity. Again, while people even then are capable of learning rules from games they have never participated in, there are visual clues and knowledge of systems that we take for granted that allow small insights that lead to an eventual unravelling of the complete rules. An example would be a New Zealand rugby fan taking an interest in an American Football (NFL) match, having never seen one before. He or she is already able to understand who is

winning overall by how many points each team has scored, who is attacking by noting which team is in control of the ball, and how well they are placed by the position they are on the field - all of which is very similar to rugby. From celebrations of players and fans and score changes the person is rapidly able to work out more specific rules, from which new rules can be induced and added.

Procedural Nature of Game Logs

If the game logs could be converted to qualitative, rather than quantitative data, it has to be noted that creating even an imitative agent using procedural rule finding would still suffer from most of the issues discussed in the previous sections. For higher level agent representations, like Quakebot, able to reason while also using mainly procedural rules, some form of architecture is necessary to combine the procedural and declarative rules. Case-based reasoning (CBR) could provide a means to side-step a need for a layered cognitive architecture, if a purely reactive replay (imitative) agent akin to Floyd et al's soccer bots (Floyd et al., 2008) is acceptable.

FPS Game Log Data not Agent-Relative

Relativity of data is an important quality for agent learning as decisions can be learnt that are generalized (e.g. enemy is to the right of agent) rather than relying on global specific (e.g. coordinates of agent vs. coordinates of enemy) data that leads to an explosion of rules that are only useful in very specific situations. However, as the data within the game logs is globalised, a significant amount of pre-processing of data using some form of perspective projection is required (see Figures 4.2 and 4.3 and Appendix A). This is effectively the difference between relative and global data; while the use of global positioning is necessary to have a general idea of the world, at the lowest level people focus on world relative to their position and bearing.

Related to this issue is the inherent map specificity of game log data that was initially noticed in the study of the TFC logs. Any decisions made by a player, even reactive ones, are likely to be affected by not only the immediate surroundings (situation specific) but also the extent of their knowledge (or belief of knowledge) of the current map (world specific) they are playing. An internal agent mapping of the world and therefore also a global positioning understanding would also be a fundamental part of any agent implementation.

Game Logs are an Unintuitive Format

Learning from game logs in some ways can be seen as a opportunistic attempt to ease the agent learning process using resources that are already available, but not widely recognised as such. In this regard, it must be noted that in both forms of game logs discussed here (parsed-for-humans and demo logs) were not built with agent learning in mind, rather as a means to fill a need in the computer game playing community.

Parsed logs were made for enthusiasts to compare statistics to their opponents to improve their own game. They do not wish to see specific examples, as they have already been through the experience, they just want a summary. Perhaps the best a modelled agent could do with these statistics is check them after every game and increase its 'difficulty' level accordingly to keep challenging the player (e.g. increase accuracy of shooting as the player improves).

Demo logs were created for video playbacks of games, again for game enthusiasts. As mentioned, for efficiency, the environmental component is not included, rather it is read from the same map files that have been built for actual use in the game itself. The demo log itself is the positioning of non-static entities around this map as viewed from a camera point for every time instance. There is no causal element in the activities of the entities: the explosion a video viewer sees as a result of a rocket is only there because at that alloted time frame and space the entity 'medium explosion' should appear there, not because the game engine has been re-used and produced it as a result of the rocket hitting the wall. Information non interesting for an agent is necessary for recreating the scene at each frame exactly, while missing information has been removed because it is surplus to requirements for the video itself. This is the reason behind the unintuitive nature of these logs, and one that is not going to be rectified until game developers see the value of such information for future research.

This means for any agent learning, a significant amount of preprocessing of data is necessary. To take an example that also affects the previous issue outlined (agent-relative data), such as learning aiming, the only information we have is the position of the player, his opponent, and the way the player is aiming his gun. While we have a lot of data for a number of different situations, the relationship between the pitch and yaw of the aimed gun and the positions of each player is trigonometric and must be calculated in a pre-processing stage before agent learning starts.

6.2 On The Use Of Rule-Based Data Mining Methods with Game Logs

Using a rule based approach to machine learning in games seemed intrinsically logical as games are created with, bounded by and structured around rules. Rules are a fundamental aspect of games. Without rules, there can be no predefined goal, and any activity without rules is perhaps better defined as 'having fun'. While rule-based data mining methods were fine on the static, declarative TFC logs, the inherent nature of the data of the demo logs meant the approach was not so suitable. Even ignoring the quantitative and procedural nature of the data, it was found the top-down, statistical nature of explicit rule induction was at odds with the bottom-up, causal requirements of the problem. Supervisory rule based data mining and subsequent implementation, while non-trivial is not interesting. It would effectively be supervisory rule mining of data and then manual hard coding of rules found.

6.2.1 Issues with Rule-Based Data Mining Methods

This section looks at the issues found in attempting to use explicit knowledge extraction methods (rule-based datamining) from the demo computer game logs. It outlines the problem of needing to know what you are looking for before starting, the issue of general concepts with low-level data, and finally the problem of rule mining and causality.

Needing To Know What to Know

The first three of the golden rules in data mining (Pyle, 1999, pg. 29) are

1. Select clearly defined problems that will yield tangible benefits.
2. Specify the required solution.
3. Define how the solution delivered is going to be used.

This poses a problem to creating an automated data mining agent, even more so given a low base level of agent intelligence to begin with (i.e. *tabula rasa*). This is particularly pertinent to rule mining due to the number of rules found by pure coincidence that have no real relevance to the aimed learning outcome. Chi-squared testing (see section 2.3) can help this issue by being able to determine if a rule is non-random. Returning

to the anecdotal example in section 6.1.2, peoples' rule mining mechanisms are similar: to learn the rules of the game from observation, we need to have some idea of what a game is, what in general a game would entail, plus some measures of behaviour leading to winning and losing play. If perhaps we had no idea at all about a game, general knowledge of points (the more the better) and perhaps celebratory or disappointed behaviour would show us at least a sequence that lead to a successful play. Case based reasoners, imitation learners and artificial neural networks can get around this problem by effectively mimicking behaviour seen. While this can be seen as robotic (in the derogatory sense of the term) and perhaps mindless (as per the old saying "You wouldn't jump off a cliff would you?") these techniques have shown to produce some results, notably with QUAKE II and neural nets (Thurau et al., 2003), genetic algorithms injected with case based instances (Louis and Miles, 2005) and simulated CBR based robocup agents (Floyd et al., 2008).

Association Rules and Causality

The issue with needing to know what to know starts before being able to address the fundamental problem of induction (whether the set of rules found predicts unforseen circumstances correctly–i.e. whether an agent using the rules found would perform well in unforseen circumstances), that is, in finding and presenting suitable examples that rule-based data mining techniques are amenable to. The next issue questions the level of supervisory interference within this process. Processes of inductance result in the designation of causality, a process that is effectively made by the trainer in statistically based data-mining methods, by providing the correct examples and ensuring rules found from those examples are valid. Rule based data mining methods use statistical correlative processes to find rules that satisfy some level of criteria (see section 2.1.4). This does not necessarily mean there is a causal element to the rule found. Causality might be a natural notion for humans (Novak and Gowin, 1984), but is a sensitive topic in statistics (i.e. *cum hoc ergo propter hoc* or 'Correlation does not imply causation'). For agent learning purposes, this not only means a decision is required as to what level of correlation determines causation in different areas of behaviour, but also being able to decide good rules from bad-impossible without expert knowledge to begin with.

On the Low-Level Nature of the Data in Game Logs

The issue of finding abstract concepts in low level data was highlighted in research by Widmer (2003) in an analogous situation to this research in attempting to find rules for expressive musical performances from performances of concert pianists. While able to find "some surprisingly simple and robust performance principles", "more abstract structural aspects... like phase structure and harmonic structure... are difficult to capture in a formal system, and we have no algorithms that could reliably compute them from the information given..."(Widmer, 2003). This underscores the inability of rule based methods to capture, in explicit terms, knowledge that is encoded implicitly within data. By contrast, in a similar study using CBR techniques, Lopez de Mantaras and Arcos (2002) presented a program that could play music expressively, having used a similar level of data to begin with. This concept is perhaps echoed in the way people learn, as Reber notes:

> "Implicit learning is the acquisition of knowledge that takes place largely independently of conscious attempts to learn and largely in the absence of explicit knowledge about what was acquired." Reber (1996, pg. 5).

Modelling Intention From many symbolic advocates' points of view, a large issue with such imitative based approaches is modelling intention. Imitative advocates counter this argument by pointing out this intention is captured implicitly in the low-level (i.e. quantitative) expert behavioural data that is being imitated. Gorman et al. (2006b) used this principle with their strategic level behavioural tier by assuming that higher level and strategy is implicit in decisions made by the imitated expert player at each item (powerup) collection point. It is a good assumption: experienced players follow paths to not only collect items but intuitively keep ideas such as powerup monopolization or collection urgency state (perhaps a closer but not so effective powerup necessary) in mind as they are doing so. It is not hard to imagine that if behaviours in these states can be followed closely by the agent, some of the intuitive behaviour inherent in powerup collection will be encompassed as well.

Differences to Expressive Musical Representation While Widmer's expressive musical performance research, nearly analogous to the current research's undertaking in finding higher order representations from base data, is inherently negative about the prospects of such an endeavour, it can be argued that computer games might be able

to overcome some of the failings of such musical analysis due to the differing intrinsic architectures of the subjects: At the highest level of a computer game the result is clear-cut: win, loss or draw, whilst judging a musical performance is still empirical rather than quantitative. Performance measures can be broken down during a game as points, or at least as positions leading to a winning position, while musical performance mid-recital is as abstract as the overall performance. However, in a similar vein it can be argued the two subjects are not so dissimilar: automated agents have been created that follow rules to competently play computer games and musical instruments, with the 'humaness' of the agent only able to be measured empirically; both areas have clearly defined rules that cannot be broken, but in a an expert's hands, these rules are interpreted into behaviour that can be perceived by even a novice as special.

6.2.2 Are Rules Appropriate in this Situation?

The original idea of using rules to define behaviours explicitly was at a semantic level, rather than a procedural one. The difference is crucial: one implies a higher level of understanding than the other. Using low-level data, even if it notes intimately every aspect of a world over a certain time frame, means without an interface (or human) to recognise semantic level events any data mining done will be at the procedural level. This has wide ranging effects that are difficult to overcome; for instance, instead of learning to shoot at an opponent, the agent can only learn to shoot around the origin of an opponent. The kind of learning is different: contextual/semantic vs. imitation.

On Procedural Rule Representation While rule learning is possible at a procedural or imitative level, its benefits are questionable - at a semantic level a simple and understandable rule will suffice to explain complex behaviours, while at a procedural level a number of rules are required to map complex behaviour, often not achieving the level of sophistication a neural network or a case based reasoner might, with less work necessary on the part of the supervisor.

Rule learning at the procedural level with no contextual references given even to humans is unintuitive. The more complex and varied the examples even with thousands of examples can lead to more, not less confusion (while it is the natural domain of neural nets and CBR systems) watching an expert perform a task and expecting to derive rules from it is even harder, as generally they would perform tasks at a faster rate, using more complex methods in more varied ways, often seamlessly linking basic skills into

more complex behaviours (conversely, with expert play neural nets and CBR systems perform even better).

On Bottom-Up Rule Learning Rule learning given context and the ability to distinguish behaviours in these situations however can be powerful. However this means that any completely automated agent learning rules from bottom-up data has to be complex enough to be able to recognise from bottom-up behaviour new (and useful) behaviour, a recording and/or imitative function to copy the behaviour, and a processing function to extract the rule from the new behaviour. Once found, the rule might have to be ranked in importance with other rules already learned, pointing towards the need for an already established architecture as well as another processing function to recognise the level the new rule should slot into.

On the Understanding Behind the Methodology The researcher's original understanding of the problem was flawed due to supposing that knowledge, broken down into fine components could be rebuilt into understandable rules that could be added to one another to gradually produce generic rules that would show deep understanding of a game, from the lowest level. While it is believed this idea is still credible, there is a level even in human understanding where the rules of the world fall away to unconscious drilled responses to situations (Schank and Abelson, 1977; Dreyfus and Dreyfus, 1980; Searle, 2004). There is a level where learned rules and automatic responses overlap - an example would be the rule that if you are learning to ride and are beginning to tip to one side of your bike you should turn your handlebars to that direction in order to regain balance. Most people who have learnt to ride a bike are probably unaware of this rule, having learnt such a response by trial and error without conscious thought. Similarly, those that were told while learning are not likely to recall that particular rule anymore either. In fact, generally once a human has learnt a task sufficiently well, the deliberative (and conscious) rule following process through habituation with the task is made automatic (Anderson, 1992).

The method of using rule-based data mining methods to extract explicit data from data logs for rules for general agents was hindered by a number of concepts:

- Game logs only really useful if the learning process is automated
- Rules are only useful if they are understandable

- Why attempt to create an agent that is worse than the current state machine/rule based bots?

6.3 Possible Approaches

Agents created to date have been either almost purely symbolic (rule-based/declarative with procedural control) or almost purely sub-symbolic (largely reactive/procedural/behaviour-based). Both approaches in using game logs as a source of expert behaviour for learning have real difficulties - generally, the inherently sub-symbolic nature of the FPS game logs (due to the inherently procedural programming of the game itself) gives rise to a number of issues for symbolic approaches, while the difficulty in producing emergent strategic level behaviour and parallelising components hangs over sub-symbolic approaches. This section shows possible approaches to implementing a means of using downloaded interactive game logs for embodied agent learning by suggesting general means of approach coupled with techniques that have overcome some of the inherent difficulties outlined above in more specific domains.

6.3.1 Symbolic/Rule-Based Approach

The largest difficulty in learning declarative rules automatically from downloaded interactive game logs is in the sub-symbolic and procedural nature of the game log data. While it could be easy to dismiss symbolic approaches altogether, the readily understandable rules and relative ease of higher level strategic planning (plus the fact games are effectively games because of the rules that make them), make them indispensable at the present time, perhaps best shown by the complex behaviours able to be implemented by Quakebot (Laird, 2001). Plus, as will be seen in the next section, for all their promise, sub-symbolic approaches have their own difficulties to overcome.

Side-Stepping the Symbol Grounding Problem

One answer (not without its own difficulties, admittedly) to the problem of the symbol grounding problem would be to make the contents of the game logs qualitative and declarative, rather than quantitative and procedural. For this to occur, it would be necessary for the game itself to work in declarative terms or at least have a declarative layer built into the game engine (Könik and Laird use a 'Symbolic Environmental

Representation' in Konik and Laird (2004)). From this layer symbolic states and transitions could be collected for game logs, to be used by symbolic cognitive architectures or integrated for implementation with artificial agent game interfaces such as Berndt et al's Open AI Standard Interface Specification (OASIS) (Berndt et al., 2005).

This layer might be built using a standardized game ontology perhaps using qualitative reasoning techniques, with particular regard to qualitative spatial reasoning (Freksa, 1991; Forbus, 1997; Guesgen, 2002; Forbus et al., 2002) (a field that uses abstractions to model physical space) and qualitative physics Forbus (1997) (a field that attempts to model the physics of a world using rules rather than exact quantitative methods). Zagal et al Zagal et al. (2005) have already begun to outline a game ontological language for game analysis that could possibly be extended to the standardised qualitative programming of some types of game, while Zhou and Ting Zhou and Ting (2006) have used qualitative methods in a three-dimensional FPS environment to model moveable objects. While these two examples, in themselves, do not come close to a complete solution to modelling a physical three dimensional simulation symbolically, they are indicative of approaches that may be successful in the future.

Using Discretisation Methods on Quantitative Data

As mentioned in section 2.2.1 there are a number of means of discretising time series data. While these do not offer solutions to the procedural nature of the data (see section 6.2.2), as discussed in section 6.3.3 below, future agents will probably require architectures with modules to deal with both procedural and declarative information.

Learning from Complex, Expert Level Logs

Learning player behaviour from game logs (even if captured declaratively) still suffers from problems outlined in section 6.1.2 above. While Nejati et al's hierarchical learning approach (Nejati et al., 2006) confronts this issue in part, like Könik and Laird's annotated expert decisions (Konik and Laird, 2004), its top down approach means that it requires positive (i.e. successful) examples with a set *goal* and *start* point for each behaviour to be learnt. As mentioned, this presents difficulties in learning from interactive game logs as it is difficult to subsume the complex and interrupted processes the recorded player takes during a standard twenty minute game into specific behaviours that can be learnt with starting and goal points.

It is believed a bottom-up approach could work better in this regard. Bottom-up approaches also useful in not requiring a certain level of agent sophistication before rule learning begins. A bottom-up approach that could be useful can be seen in Rauterberg (1996), using Petri Nets in task planning, though not in such a complex environment. Bottom up approaches still require a means of hierarchical rule classification, meaning some level of basic cognitive architecture is probably necessary, even if a standard interface is used.

General Issues with Using a Symbolic Approach

Using abstractions to define the world comes at the price of losing some information. Some of the difference between beginner and expert players could well be encoded in the fine motor control of the player characters, information that could be lost in creating a symbolic layer. A lack of granularity at an abstract level could be an important aspect of an inability to create realistic and expert level agents.

6.3.2 Sub-Symbolic Approaches

A sub-symbolic approach to modelling an agent would seem to be the best way to approach the problem given the nature of the data in the current game logs. To emphasise this point, to this stage it has been the only approach in using game logs to learn from in creating a FPS agent (e.g. Gorman et al. (2006a)). Perhaps more efforts have not been made in this area by sub-symbolic and nouvelle AI advocates due to a generally held belief championed by Brooks (1990) condemning simulated environments. Two large stumbling blocks for sub-symbolic approaches have been the creation of emergent behaviours that match the higher level thought processes common to symbolic approaches and parallelisation of multiple task modules into a coherent whole. It is interesting to note that after beginning with completely reactive, strictly bottom up approaches using neural nets and self organising maps to model agents' movement Thurau et al. (2003), later strategic layered approaches took on more top-down attributes including setting nodal positional points (formed by clustering the entire set of player positions) which the agent would travel between and the making goals in the form of item pickup points Gorman et al. (2006a).

Preprocessing of Data

Using quantitative rather than qualitative data could be seen as returning to the original premise of attempting to model a complete agent: to be forced to create sub-cognitive means to deal with data that is much closer to the kind of data a real-world agent would face. However, issues such as overabundance of data and non agent-relative data (see section 6.1.2) mean a large amount of preprocessing of the data is required, even if specific logs are created (instead of using expert logs from the online repositories) before it can be used for learning. While it could be argued that the important part of the logs is the behavioural data they contain, not the form they are in, the ability to 'cheat' (whether consciously or unconsciously) to produce circumstances favourable to the agent's learning circumstances is possible. Means to extract important features or bring levels of dimensionality down with large amounts of data are self-organised maps, principle component analysis (both used in methods by Thurau et al. (2003, 2005)) and independent component analysis. Until data is presented in an agent relative fashion, or video images are used, it is difficult to see a means around manipulating data using perspective projection techniques (see section 5.2.6). The success of Floyd et al. (2008) in largely automating the learning process, even in a less complex environment and producing effectively reactive agents, shows CBR methods could provide a means to sidestep this process.

Case-Based Reasoning Approach

It should be possible to mimic the advanced ambush play of Quake II experts using a case based reasoner. Using an ordinary waypoint system for traversing the map and collection of useful items (better weapons and ammunition, extra armour, etc) combined with a case based reasoner interrupt program triggered by events such as seeing an enemy or being shot at. The cases, or instances would be a series of actions made by expert players in that similar position over several seconds. Speed of finding the correct case would be aided by having ready a narrowed set of instances while wandering searching for an enemy, chosen and updated as the agents' level of health, weapons and level of ammunition, and location change. An instance of that narrowed set could be chosen once the enemy has been seen, noting his/her location and weaponry. As has been mentioned in section 6.1.2, this approach would be entirely reactive unless some form of higher implicit intentionality could be captured. A similar example to

this was seen in a Robocup environment with Ros et al. (2007), who used sequences of events to build team play amongst Robocup agents.

Other more complicated means could emulate disembodied agent examples in strategic game settings such as Mayor (Fasciano, 1996) with multiple agents updating a task list that a central processing system prioritised or the integration of other machine learning methods as CIGARS (Louis and Miles, 2005) was able to do. As mentioned in section 6.1.2, another means would be to include a declarative reasoning module, though of course this is dependent on the original data being of a qualitative nature.

Other Imitative Approaches

Gorman et al's version of capturing the implicit intentionality of human players was based on assuming the actions the players chose between each inventory state and item collection point had a higher strategic component embedded within it. As an aside, while it is recognized this implementation was far from complete, there are some weaknesses with this approach, most obviously being the lack of recognition given to the importance of opponent behaviour in making item collection decisions. This particularly affects the fluidity of the implicit abstract strategic behaviours between collections of powerups (by collecting an item, the agent's state changes and a new path is decided upon, regardless of the situation the observed player was in at the time).

While the fact the agent performed strongly in the group's believability test is indicative of the potential of this approach, to compare the imitative agent to a rule-based agent and draw unfavourable comparisons is questionable. Firstly, the rule-based agent being compared has been modelled as a complete game player; its strengths perhaps lie in its generality and ability to play all parts of the game including crucially against opposition. By comparison, the imitative agent has effectively been specifically engineered to this task. Secondly, symbolic approaches, being generally top-down approaches, do not give the same level of emphasis (if any at all) to the authenticity of lower-level actions, concentrating on producing higher level tactical planning by modelling human thought processes, such as ambush behaviour (Laird and van Lent, 2000).

However, the comparison here was not for overall effectiveness or playability but for a level of 'humaness'. Indeed, the rule-based bot, which the imitative agent was compared with, had been "Highly recommended for it's (sic) human like AI" (MrFreeze, 1999). Incidentally, Laird and Duchi (2000) found that in their own 'humaness' tests for Quakebot (mentioned earlier), major factors in believability came down to aiming

accuracy and decision time (i.e. procedural actions), while skill level was not such a factor.

6.3.3 Hybrid Agents

Whether Gorman et al's agent's emergent behaviour from Bayesian imitation of player movements extended past the twenty second believability tested time intervals (Gorman et al., 2006b) is not obvious. However, it was only through the creation of what were effectively clustered abstractions (see section 3.3.2) from custom logs that the group were able to integrate higher level strategic planning into their agent. It is interesting that the group followed this line of research leading them to the borderline of symbolic approaches rather than continuing on the more favoured behavioural sub-symbolic approach of parallelisation of numerous behavioural modules to produce emergent behaviour.

It is difficult to see FPS games changing any time soon to a declarative programming process. In many ways, as already mentioned in section 6.3.2, using quantitative data more stays truer to the form of data agents would receive from sensors in the real world. As section 6.1.1 has suggested, perhaps the greatest motivation for modelling in simulated, situated, three dimensional environments like FPS game worlds is their closeness to reality, without the noise issues. This would point to using design approaches that combined symbolic and sub-symbolic approaches (so called 'hybrid' agents). The ICARUS cognitive architecture (Langley et al., 1991) already has some features to this effect. Also interesting to note is the sub-symbolic module that has being added to the Soar architecture quite recently Laird (2008).

From a pragmatist's view it could be said that taking features from both symbolic and sub-symbolic approaches could lead to an approach that melds the best features of both. However, the two approaches are largely dischordant in a number of ways and their base philosophical backgrounds are largely at odds. Explicitly, is the hybrid agent being created going to model a human by recreating the cognitive process to create behaviours, or is it being created to mimic very closely the actual behaviour of a human, giving the illusion of an intelligent agent? Perhaps the idea is not so black and white, as it seems humans need both kinds of learning (Choi et al., 2007). While people make rational decisions to govern many actions, there are a number of ticks inherent in our behaviour that serve no logical purpose, but make up a narrative component for others to pick up on, that becomes part of the way people communicate, makes

people part of a group, also giving insights to others as to what people are thinking. These small ticks or behaviours come about from the copying of parents or companions and they mark people as a member of a certain group (Rao et al., 2004). Perhaps this narrative component is the same narrative component Sengers (2002) was talking about (see section 6.1.1) that has been missing from computer game agents.

Chapter 7

Conclusions

Downloaded, expert-level, interactive computer demo game logs show promise as a future source for expert behaviour to supplement or even replace time-consuming supervised expert training schemes. However, this promise is currently balanced by a number of issues that need to be addressed before their full potential can be utilised. Possible approaches are thought to exist to use computer game logs for agent learning, though it is uncertain whether the effort of pre-processing of data that is required for any approach taken exceeds the convenience and content factors gained in the use of the logs.

7.1 Main Conclusions

Are downloadable game logs useful for agent learning? While there is large potential, for a number of reasons, for the use of downloaded computer game logs for agent learning, it has to be said that the level of the usefulness of the thousands of interactive logs in online repositories at the present time to model embodied agents is small, especially for symbolic approaches.

Main Issue Both the content and means to extract that content automatically has a large bearing on their usefulness: the larger the supervisory aspect required by the method of agent training, the less useful they become, as it is easier to produce single logs and elucidate every action separately in (basic) movements for agents to learn than to extract behaviours from the thousands of game logs created by expert players by hand.

On using rule-mining techniques The top-down, statistical nature of explicit rule induction is at odds with the bottom-up, causal requirements of the problem.

General Recommendations Recommendations to increase this usefulness can be generalised into two parts: firstly, the improvement of means of information representation in the logs themselves; and secondly, the choice of methods by the researcher that lead to largely automatic information extraction.

7.2 Why Game Logs have the Potential as a Source for Agent Learning

The environmental and behavioural data that is captured within the demo log files and their corresponding map files is the main drawcard for using interactive computer game logs for agent learning purposes.

- Complex, realistic, interactive, accesible and unique testbed for situated and embodied agents.

The behaviour captured within the demo logs is unique for being spread over so many different players at an expert level, a variety and expertise that cannot be matched by a researcher creating behaviour from scratch.

- Behaviour captured has a motivated element, is expert-level and includes cognition at reactive, tactical and strategic levels.
- The simplicity of the game belies nuanced ways to approach the game. Expert players tend to defensive and ambush tactics.
- A fine grained level of behaviour means capture of intuitively human behaviours. Potential for opponent modelling evident.

Game logs have an inherent convenience factor.

- No effort to create, small effort to download.
- Possibility for offline learning.

7.3 General Issues with Game Logs for Agent Learning

Sparsity/Overabundance of Data

- Parsed logs for human consumption too sparse.
- Demo logs mean difficulties in filtering data to find the correct data.

Symbol Grounding Problem

- Quantitative representation of data means assignment of symbols to data is difficult.
- Temporal rule mining discretisation and labelling methods often require sequential and pre-defined symbols.

Continuous Nature of Data

- Learning from a sequence of expert moves requires expert knowledge to break these sequences into basic moves.

Background Knowledge

- *Tabula rasa* or from scratch agent learning (bootstrapping) is a difficult process that requires a high granularity of training data and highly supervised or largely imitative learning processes.

Procedural Nature of Game Logs

- The procedural nature of the data means the path taken by the player through the game is the only path to learn from.
- Even with other logs and examples, decisions for specific instances are limited by this fact.

FPS Game Log Data not Agent-Relative

- A significant amount of pre-processing of data using some form of perspective projection is required.

Game Logs are an Unintuitive Format

- Game logs were not built with agent learning in mind, rather as a means to fill a need in the computer game playing community.
- Any approach taken will need to pre-process many aspects of the data before it is ready for agent learning.

7.4 General Issues with Rule-Based Data Mining Methods on Game Logs for Agent Learning

Needing To Know What to Know.

- A fundamental component of data mining methodology is knowing what is being looked for.
- For game logs to remain useful for data mining, they must hold their convenience factor. This is destroyed if every action must be elucidated by the trainer.

Association Rules and Causality.

- Correlation does not imply causation.

Low-Level Nature of Data within Game Logs.

- Implicit knowledge cannot be found using explicit methods.

Are Rules Appropriate in the Current Situation?

- No, not with the logs in the current state:
  - Game logs only really useful if the learning process is automated;
  - Rules are only useful if they are understandable;
  - Why attempt to create an agent that is worse than the current state machine/rule based bots?

7.5 Possible Approaches

Symbolic:

- Qualitative and declarative programming of game logs necessary using a standard ontology.
- Standard game AI interface.
- Bottom-up learning of hierarchical rule structure recommended.

Subsymbolic:

- Largely imitative processes probably most useful.
- Sub-symbolic methods closer to real-world elements in situatedness, embodiment and sensory modelling.
- Perhaps a combination of CBR and other machine learning processes could break the purely reactive agent barrier.

Hybrid:

- A combination of the two approaches could meld the best features of both.

7.6 Two Questions for Future Researchers

For future researchers contemplating using interactive computer game logs for general agent research, two main questions must be asked of themselves before starting. The first question, which greatly affects the second, is:

> Which philosophical approach is going to be taken?

The second, though dependent on the first question, is felt to be the more important of the two:

> Do the motivations for using game logs for agent learning make up for the amount of pre-processing of data that will be required for this approach?

These two questions and the major question asked at the beginning of the thesis can be answered simply by ensuring the following equation holds true:

> *Benefits of gamelogs > Issues of the approach that need to be overcome*

If not, then either a different approach is required, or computer game logs will not suit the needs of the research.

Appendix A

Perspective Projection Formula

$\mathbf{a}_{x,y,z}$ – *the 3D point to be projected to 2D*
$\mathbf{c}_{x,y,z}$ – *the camera location*
$\theta_{x,y,z}$ – *the camera rotation*
$\mathbf{e}_{x,y,z}$ – *the position of the viewer in camera space*

$\mathbf{b}_{x,y}$ – *the final 2D projection of* $\mathbf{a}_{x,y,z}$
$\mathbf{d}_{x,y,z}$ – *translation of point* $\mathbf{a}_{x,y,z}$ *into coordinate system defined by* $\mathbf{c}_{x,y,z}$
Where:

$$\mathbf{d}_x = cos\theta_y \cdot (sin\theta_z \cdot (\mathbf{a}_y - \mathbf{c}_y) + cos\theta_z \cdot (\mathbf{a}_x - \mathbf{c}_x)) - sin\theta_y \cdot (\mathbf{a}_z - \mathbf{c}_z)$$

$$\begin{aligned}\mathbf{d}_y = sin\theta_x \cdot (cos\theta_y \cdot (\mathbf{a}_z - \mathbf{c}_z) + sin\theta_y \cdot (sin\theta_z \cdot (\mathbf{a}_y - \mathbf{c}_y) + cos\theta_z \cdot (\mathbf{a}_x - \mathbf{c}_x))) \\ + cos\theta_x \cdot (cos\theta_z \cdot (\mathbf{a}_y - \mathbf{c}_y) + sin\theta_z \cdot (\mathbf{a}_x - \mathbf{c}_x))\end{aligned}$$

$$\begin{aligned}\mathbf{d}_z = cos\theta_x \cdot (cos\theta_y \cdot (\mathbf{a}_z - \mathbf{c}_z) + sin\theta_y \cdot (sin\theta_z \cdot (\mathbf{a}_y - \mathbf{c}_y) + sos\theta_z \cdot (\mathbf{a}_x - \mathbf{c}_x))) \\ - sin\theta_x \cdot (cos\theta_z \cdot (\mathbf{a}_y - \mathbf{c}_y) + sin\theta_z \cdot (\mathbf{a}_x - \mathbf{c}_x))\end{aligned}$$

and thus:

$$\mathbf{b}_x = (\mathbf{e}_x - \mathbf{d}_x)(\mathbf{e}_z/\mathbf{d}_z)$$

$$\mathbf{b}_y = (\mathbf{e}_y - \mathbf{d}_y)(\mathbf{e}_z/\mathbf{d}_z)$$

(Wikipedia, 2008)

Bibliography

Adobbati, R., Marshall, A. N., Scholer, A., and Tejada, S. Gamebots: A 3d virtual world test-bed for multi-agent research. In Wagner, T. and Rana, O. F., editors, *Proceedings of the Second International Workshop on Infrastructure for Agents, MAS, and Scalable MAS*, 2001. URL: `citeseer.ist.psu.edu/477805.html`.

Agrawal, R., Imielinski, T., and Swami, A. Mining association rules between sets of items in large databases. In *Proceedings of the 5th International Conference on Management of Data*. ACM Press, 1993.

Agrawal, R., Psaila, G., Wimmers, E. L., and Zaït, M. Querying shapes of histories. In *VLDB '95: Proceedings of the 21th International Conference on Very Large Data Bases*, pages 502–514, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc., 1995.

Agrawal, R. and Srikant, R. Mining sequential patterns. In *ICDE95: Proceedings of the 11th International Conference on Data Engineering*, 1995.

Agre, P. E. and Chapman, D. Pengi: An implementation of a theory of activity. In Forbus, K. and Shrobe, H. E., editors, *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 268–272. Morgan Kaufmann, 1987.

Anderson, J. R. Automaticity and the act* theory. *American Journal of Psychology*, 105(2):165–180, 1992.

Anderson, J. R. and Lebiere, C. *The Atomic Components of Thought*. Erlbaum, Mahwah, NJ, 1998.

Antunes, C. M. and Oliveira, A. L. Temporal data mining: An overview, 2001.

Bates, J. Virtual reality, art and entertainment. *The Journal of Teleoperators and Virtual Environments*, 1(1):133–138, 1991.

Bauckhage, C., Gorman, B., Thurau, C., and Humphrys, M. Learning human behavior from analyzing activities in virtual environments. *MMI-Interaktiv*, 12:3–17, 2007.

Bauckhage, C. and Thurau, C. Towards a fair n square aimbot using mixtures of experts to learn context aware weapon handling. In *Proceedings of Game-On '04*, pages 20–24, 2004.

Bauckhage, C., Thurau, C., and Sagerer, G. Learning human-like opponent behavior for interactive computer games. In Michaelis, B. and Krell, G., editors, *Pattern Recognition*, volume 2781, page 148155. Springer-Verlag, Berlin Heidelberg, 2003.

Berndt, C. N., Watson, I., and Guesgen, H. Oasis: An open ai standard interface specification to support reasoning, representation and learning in computer games. In Aha, D., Muoz-Avila, H., and van Lent, M., editors, *Proceedings of the 2005 IJCAI Workshop on Reasoning, Representation, and Learning in Computer Games*, pages 19–24, 2005.

Best, B. J. and Lebiere, C. Spatial plans, communication, and teamwork in synthetic mout agents. In ., editor, *Proceedings of the Twelfth Conference on Behavior Representation In Modeling and Simulation*, 2003.

Best, B. J., Lebiere, C., and Scarpinnatto, K. C. Modeling synthetic opponents in mout training simulations using the act-r cognitive architecture. In *Proceedings of the 11th Conference on Computer Generated Forces and Behavior Representation*, 2002.

Bhandari, I., Colet, E., Parker, J., Pines, Z., Pratrap, R., and Ramanujam, K. Advanced scout: Data mining and knowledge discovery in nba data. *Data Mining and Knowledge Discovery*, 1:121125, 1997.

Blargh. The blarghalyzer. (2005). URL: `http://www.blarghalyzer.org/` [last checked: 10 August 2007].

Brooks, R. A. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14–23, 1986.

Brooks, R. A. Elephants don't play chess. *Robotics and Autonomous Systems*, 6(1):3–15, 1990.

Brooks, R. A. New approaches to robotics. *Science*, 253:1227–1232, 1991.

Brooks, R. A. Planning is just a way of avoiding figuring out what to do next. In *Cambrian Intelligence : The Early History of the New AI*, pages 103–110. MIT Press, Cambridge, Mass, 1999.

Champandard, A. Fear sdk. (2002). URL: `http://fear.sourceforge.net`.

Chi, M. T. and Ohlsson, S. *Cambridge Handbook of Thinking and Reasoning*, chapter Complex Declarative Learning. Cambridge University Press, 2005.

Choi, D., Konik, T., Nejati, N., Park, C., and Langley, P. A believable agent for first-person perspective games. In *Proceedings of the Third Artificial Intelligence and Interactive Digital Entertainment International Conference (AIIDE 2007)*. AAAI Press, 2007.

Cohen, W. W. Fast effective rule induction. In *Proceedings of the 12th International Conference on Machine Learning (ICML'95)*, pages 115–123, 1995.

Crandall, R. W. and Sidak, J. G. Video games: Serious business for america's economy. Technical report, Entertainment Software Association, 2006.

Das, G., Lin, K.-I., Mannila, H., Renganathan, G., and Smyth, P. Rule discovery from time series. In *KDD*, pages 16–22, 1998.

Daw, C. S., Finney, C. E. A., and Tracy, E. R. A review of symbolic analysis of experimental data. *Review of Scientific Instruments*, 74(2):915–930, 2003.

dethkultur, Hoony, DonKing, and Agent. Challenge tv. (2000). URL: `http://challenge-tv.com/` [last checked: 7 May 2008].

Dreyfus, H. and Dreyfus, S. *Advances in Cognitive Science 1*, chapter Why Skills Cannot Be Represented by Rules, pages 315–335. Ellis Horwood Ltd., 1980.

Epic Games,. and Digital Extremes,. *Unreal Tournament. (GT Interactive)*. [Computer Software], 1999.

Fasciano, M. J. Real-time case-based reasoning in a complex world. Technical report, Computer Science Department, University of Chicago, 1996.

Fielding, D., Fraser, M., Logan, B., and Benford, S. Extending game participation with embodied reporting agents. In *Proceedings of the ACM SIGCHI International*

Conference on Advances in Computer Entertainment Technology, volume 74, pages 100 – 108. ACM Press, 2004.

Finney, S., Gardiol, N. H., Kaelbling, L. P., and Oates, T. Learning with deictic representation. Technical report, MIT: http://hdl.handle.net/1721.1/6685, 2002. URL: `http://hdl.handle.net/1721.1/6685`.

Floyd, M. W., Esfandiari, B., and Lam, K. A case based reasoning approach to imitating robocup players. In *Proceedings of the Twenty-First International FLAIRS Conference*. AAAI Press, 2008.

Forbus, K. D. Qualitative reasoning. In Tucker, J. A., editor, *The Computer Science and Engineering Handbook*, pages 715–733. CRC Press, Boca Raton, FL, 1997.

Forbus, K. D., Mahoney, J. V., and Dill, K. How qualitative spatial reasoning can improve strategy game ais. *IEEE Intelligent Systems*, 17(4):25–30, 2002.

Freksa, C. Qualitative spatial reasoning. In Mark, D. M. and Frank, A., editors, *Cognitive and Linguistic Aspects of Geographic Space*, pages 361–372. Kluwer Academic Publishers, Dordrecht, 1991.

Girlich, U. The unofficial dm2 format description. (2000). URL: `http://demospecs.planetquake.gamespy.com/dm2/dm2.html` [last checked: 2 July 2008].

Gorman, B., Fredriksson, M., and Humphrys, M. Qase - an integrated api for imitation and general ai research in commercial computer games. In *Proceedings of the IEEE 7th International Conference on Computer Games*, pages 207–214. CGAMES, 2005.

Gorman, B. and Humphrys, M. Towards integrated imitation of strategic planning and motion modeling in interactive computer games. *Comput. Entertain.*, 4(4):10, 2006.

Gorman, B. and Humphrys, M. Imitative learning of combat behaviours in first-person computer games. In *Proceedings of the 10th International Conference on Computer Games: AI, Mobile, Educational & Serious Games*, 2007.

Gorman, B., Thurau, C., Bauckhage, C., and Humphrys, M. Bayesian imitation of human behavior in interactive computer games. In *Proceedings of the 18th Inter-*

national Conference on Pattern Recognition, volume 1, pages 1244–1247. ICPR, 2006a.

Gorman, B., Thurau, C., Bauckhage, C., and Humphrys, M. Believability testing and bayesian imitation in interactive computer games. In *Proceedings of the 9th International Conference on the Simulation of Adaptive Behavior (SAB06)*, pages 655–666. Springer, 2006b.

Guesgen, H. Reasoning about distance based on fuzzy sets. *Applied Intelligence*, 12(3):265–270, 2002.

Harnad, S. The symbol grounding problem. *Physica*, D 42:335:346, 1990.

Henry, C. and Karsemeyer, J. Physics in mass market games. (2008). URL: `http://www.gamecareerguide.com/features/534/features/534/physics_in_mass_market_.php` [last checked: 9 July 2008].

Hollnagel, E. *Human Reliability Analysis: Context & Control*. Computers and People. Academic Press, London, 1993.

IBM Research, . Slam - semantic learning and analysis of multimedia. (2008). URL: `http://www.research.ibm.com/slam/` [last checked: 9 July 2008].

id Software Inc. *Quake II*. Activision, Hyperion Entertainment (Amiga). [Computer Software], 1997.

id Software Inc. *Quake III Arena*. Activision, Sega. [Computer Software], 1999.

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., and Hinton, G. E. Adaptive mixture of local experts. *Neural Computation*, 3:79–87, 1991.

Khoo, A., Dunham, G., Trienens, N., and Sood, S. Efficient, realistic npc control systems using behavior-based techniques. In *Proceedings of the AAAI 2002 Spring Symposium Series: Artificial Intelligence and Interactive Entertainment*, 2002.

King, R. D., Srinivasa, A., and Dehaspe, L. Warmr: a data mining tool for chemical data. *Journal of Computer-Aided Molecular Design*, 15(2):173–181, 2001.

Konik, T. and Laird, J. Learning goal hierarchies from structured observations and expert annotations. In *Proceedings of the Fourteenth International Conference on Inductive Logic Programming*, pages 198–215. Springer, 2004.

Kuhlmann, G., Knox, W. B., and Stone, P. Know thine enemy: A champion robocup coach agent. In *In Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI 06)*, page 146368. AAAI, 2006.

Laird, J. Soar tutorial part 6. (2006). URL: `http://ai.eecs.umich.edu/soar/sitemaker/docs/tutorial/TutorialPart6.pdf` [last checked: 27 June 2008].

Laird, J. Extending the soar cognitive architecture. (2008). URL: `http://ai.eecs.umich.edu/people/laird/papers/Laird-GAIC.pdf` [last checked: 28 June 2008].

Laird, J. E. It knows what you're going to do: adding anticipation to a quakebot. In *AGENTS '01: Proceedings of the fifth international conference on Autonomous agents*, pages 385–392, New York, NY, USA. ACM, 2001.

Laird, J. E. and Duchi, J. C. Creating human-like synthetic characters with multiple skill levels: A case study using the soar quakebot. In *Proceedings of the AAAI Fall Symposium: Simulating Human Agents*. AAAI Press, 2000.

Laird, J. E., Newell, A., and Rosenbloom, P. S. Soar: An architecture for general intelligence. *Artificial Intelligence*, 33:1–64, 1987.

Laird, J. E. and van Lent, M. Developing an artificial intelligence engine. In *Proceedings of the Game Developers Conference, San Jose, CA*, pages 577–588, 1999.

Laird, J. E. and van Lent, M. Human-level ais killer application: Interactive computer games. In *Proceedings of the AAAI Fall Symposium*, pages 80–97. AAAI Press, 2000.

Langley, P., McKusick, K., Allen, J., Iba, W., and Thompson, K. A design for the icarus architecture. *SIGART Bulletin*, 2:104–109, 1991.

Lattner, A. D., Miene, A., Visser, U., and Herzog, O. Sequential pattern mining for situation and behavior prediction in simulated robotic soccer. In *RoboCup*, pages 118–129, 2005.

Lavrac, N. and Dzeroski, S. *Inductive Logic Programming: Techniques and Applications*. Ellis Horwood, Chichester, 1994.

Lionhead Studios,. *Black & White*. EA Games. [Computer Software], 2001.

Lopez de Mantaras, R. and Arcos, J. L. Ai and music: From composition to expressive performance. *AI Magazine*, 23(3):43–58, 2002.

Louis, S. J. and Miles, C. Combining case-based memory with genetic algorithm search for competent game ai. In *ICCBR Workshops*, pages 193–205, 2005.

Maxis. *The Sims*. EA Games. [Computer Software], 2000.

McCarthy, J. Partial formalizations and the lemmings game. (1998). URL: `http://www-formal.stanford.edu/jmc/lemmings.html` [last checked: 13 May 2008].

McGrath, D. and Hill, D. Unrealtriage: A game-based simulation for emergency response. In *Huntsville Simulation Conference*, 2004.

Millington, I. *Artificial Intelligence for Games*. Series in Interactive 3D Technology. Elsevier, San Francisco, CA, 2006.

Mitchell, T. M. *Machine Learning*. McGraw-Hill, New York, 1997.

Molineaux, M. and Aha, D. Tielt: A testbed for gaming environments. In *Proceedings of the 16th National Conference on Artificial Intelligence*, pages 1690–1691, 2005.

Monolith Productions,. *F.E.A.R.* Vivendi Universal. [Computer Software], 2005.

MrFreeze. Reviews - gladiator bot. (1999). URL: `http://leesvoer.com/zooi/projects/gladiator/reviews.html` [last checked: 9 July 2008].

Muggleton, S. Inverse entailment and progol. *New Generation Computing*, 13:245–286, 1995.

Nejati, N., Langley, P., and Konik, T. Learning hierarchical task networks by observation. In *Proceedings of the 23 rd International Conference on Machine Learning*, volume 148, pages 665–672. ACM Press, 2006.

Newell, A. *Unified Theories of Cognition*. Harvard University Press, Cambridge, Massachusetts, 1990.

Novak, J. and Gowin, D. B. *Learning to Learn*. Cambridge University Press, 1984.

Olsen, S. Humans best computer in poker match. (2007). URL: `http://www.news.com/8301-10784_3-9750210-7.html` [last checked: 13th Sept 2007].

Pfeifer, R. and Scheier, C. *Understanding Intelligence*. MIT Press, Cambridge, MA, USA. Illustrator-Isabelle Follath, 2001.

Poole, D., Mackworth, A., and Goebel, R. *Computational Intelligence: A logical approach*. Oxford University Press, New York, 1998.

Pyle, D. *Data Preparation for Data Mining*. Morgan Kaufmann, San Francisco, CA, 1999.

Quinlan, J. Learning logical definitions from relations. *Machine Learning*, 5(3):239–266, 1990.

Rao, R. P. N., Shon, A. P., and Meltzoff, A. N. A bayesian model of imitation in infants and robots. In Dautenhahn, K. and Nehaniv, C., editors, *Imitation and Social Learning in Robots, Humans, and Animals: Behavioural, Social and Communicative Dimensions*. Cambridge University Press, 2004.

Rareware. *Goldeneye 007*. Nintendo. [Computer Software], 1997.

Rauterberg, M. A petri net based analysing and modelling tool kit for logfiles in human-computer interaction. In Yoshikawa, H. and Hollnagel, E., editors, *Proceedings of Cognitive Systems Engineering in Process Control: CSEPC'96*, pages 268–275. Kyoto University: Graduate School of Energy Science, 1996.

Reber, A. S. *Implicit Learning and Tacit Knowledge: An Essay on the Cognitive Unconscious*. Oxford University Press, New York, 1996.

Riddle, P., Segal, R., and Etzioni, O. Representation design and brute-force induction in a boeing manufactoring domain. *Applied Artificial Intelligence*, 8:125–147, 1994.

Ros, R., de Mántaras, R. L., Arcos, J. L., and Veloso, M. M. Team playing behavior in robot soccer: A case-based reasoning approach. In *ICCBR*, pages 46–60, 2007.

Russell, S. J. and Norvig, P. *Artificial Intelligence: A Modern Approach*. Prentice Hall Series in Artificial Intelligence. Prentice Hall, New Jersey, 2nd edition, 2003.

Sacchi, L., Larizza, C., Combi, C., and Bellazzi, R. Data mining with temporal abstractions: learning rules from time series. *Data Mining Knowledge Discovery*, 15:217–247, 2007.

Schaeffer, J. The games computers (and people) play. In *Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, 2000.

Schank, R. and Abelson, R. *Scripts, plans, goals, and understanding.* Lawrence Erlbaum Assoc., Hillsdale, NJ, 1977.

Schreiner, T. Artificial intelligence in game design. (2002). URL: `http://ai-depot.com/GameAI/Design.html` [last checked: 12 June 2008].

Searle, J. *Mind: A Brief Introduction.* Oxford University Press, 2004.

Segal, R. and Etzioni, O. Brute home page. (1997). URL: `http://members.aol.com/richsegal/brute/` [last checked: 12 June 2008].

Sengers, P. Schizophrenia and narrative in artificial agents. *Leonardo*, 35:427–431, 2002.

Shahar, Y. A framework for knowledge-based temporal abstraction. *Artificial Intelligence*, 90(1-2):79–133, 1997. URL: `citeseer.ist.psu.edu/shahar97framework.html`.

Thurau, C., Bauckhage, C., and Sagerer, G. Combining self organizing maps and multilayer perceptrons to learn bot-behavior for a commercial game. In *Proceedings of GAME-ON03*, pages 119–123, 2003.

Thurau, C., Bauckhage, C., and Sagerer, G. Learning human-like movement behavior for computer games. In *SAB04: Proceedings of the 8th International Conference on the Simulation of Adaptive Behavior*, 2004a.

Thurau, C., Bauckhage, C., and Sagerer, G. Synthesizing movements for computer game characters. In Rasmussen, C., Blthoff, H., Schlkopf, B., and Giese, M., editors, *Pattern Recognition*, volume 3175, pages 179–186. Springer, Berlin / Heidelberg, 2004b.

Thurau, C., Paczian, T., and Bauckhage, C. Is bayesian imitation learning the route to believable gamebots? In *GAME-ON North America*, pages 3–9, 2005.

Thurau, C., Sagerer, G., and Bauckhage, C. Imitation learning at all levels of game-ai. In *Proceedings of the International Conference on Computer Games, Artificial Intelligence, Design and Education*, page 402408, 2004c.

Tveit, A. and Tveit, G. B. Game usage mining: Information gathering for knowledge discovery in massive multiplayer games. In *Proceedings of the International Conference on Internet Computing, Session on Web Mining*, pages 636–642. CSREA Press, 2002. URL: `citeseer.ist.psu.edu/tveit02game.html`.

Valve Software,. *Half-Life*. Sierra Studios, Electronic Arts & Valve Software. [Computer Software], 1998.

Weizenbaum, J. Eliza. *Communications of the ACM*, 9:36–45, 1966.

Wender, S. Data mining and machine learning with computer game logs. Post grad project, University of Auckland, Auckland, New Zealand, 2007.

Widmer, G. Discovering simple rules in complex data: A meta-learning algorithm and some surprising musical discoveries. *Artificial Intelligence*, 146:129148, 2003.

Wikipedia. 3d projection. (2008). URL: `http://en.wikipedia.org/wiki/Perspective_projection#Perspective_projection` [last checked: 28 June 2008].

Witten, I. H. and Frank, E. *Data Mining: practical Machine Learning Tools and Techniques*. The Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, San Francisco, CA, 2nd edition, 2005.

Wray, R. E., Laird, J. E., Nuxoll, A., Stokes, D., and Kerfoot:, A. Synthetic adversaries for urban combat training. *AI Magazine*, 26:82–92, 2005.

Zagal, J., Mateas, M., Fernandez-Vara, C., Hochhalter, B., and Lichti, N. Towards an ontological language for game analysis. In *Proceedings of the Digital Interactive Games Research Association Conference (DiGRA 2005)*, 2005.

Zhou, S. and Ting, S.-P. Qualitative physics for movable objects in mout. In *ANSS '06: Proceedings of the 39th annual Symposium on Simulation*, pages 320–325, Washington, DC, USA. IEEE Computer Society, 2006.