

# Six Month Report

**Mark A. Wood (cspmaw@cs.bath.ac.uk)**  
University of Bath; Department of Computer Science  
Bath, England BA2 7AY; United Kingdom

May 5, 2004

## Abstract

This report is a review of the work I have done during the first six months of my PhD, and a research proposal for the remainder.

## 1 Introduction

I have tried to keep the two middle review sections as formal as possible, so they would require a minimum of reworking if I wanted to use them in my dissertation. This being the case, the surrounding sections are somewhat more informal, describing the ‘whens’ and ‘whys’ of my work, with ‘whats’ at only a high level. I hope this hasn’t resulted in too disjoint a style of report.

### 1.1 Road Map

Before I begin the report proper, I will use this section to set out its structure and contents for ease of reference. For the remainder of this introduction, I will explain something about my academic background, experiences leading up to joining the university and work completed over the last six months. The following section clarifies my chosen research area and justifies that choice.

Then follow my review sections: first, a literature review covering the three main topics of my research so far, and focusing particularly on computer game AI. Second, a systems review which describes the software I have used thus far, what I have used it for and what (if anything) I intend to use it for in the future. Finally, I outline my plans for the next six months and the rest of my PhD, and conclude with a short summary.

### 1.2 Background and Motivations

To best make sense of the past six months’ work, it should be helpful to look over some of the history motivating that work.

My first degree was in Mathematics with Computing, studying at the University of Warwick. In my final year I took computer science courses in graphics, robot science and algorithmic complexity (as well as maths courses in topology, knot theory, fractal geometry, algebra, group theory and history). At the time I applied to Bath, my interest was in robotics, and to a lesser degree, graphics, and the geometry inherent to both fields.

Upon enquiring to the department about studying robotics at Bath, I was directed to Dr. Nicolai Vorobjov. He told me that to say he was a robotics expert was a slight exaggeration, but he had done work in mapping topological spaces to robot configuration spaces. At that point I wasn't sure how theoretical a direction I wanted to take, so I made some further enquiries and was subsequently introduced to Dr. Joanna Bryson. Up to that point, the only AI I had come across was a bit of neural network theory.

She explained a bit about her research to me by e-mail, and pointed me in the direction of her publications web page, instructing me to "read a few papers that sound interesting" (see Section 3.1). At first, I simply read those with the most engaging title/abstract, then those which explained a bit more of the background and theory behind these, and finally her PhD dissertation (Bryson, 2001). By this time I was really beginning to develop an interest in AI, but was also intrigued by another problem I had discovered during this initial contact period: the motion blending problem (see Section 3.2). This appealed to me mainly because it was defined in terms of geometry and computer graphics; areas I was more familiar with at the time.

### 1.3 Timetable

This section gives a sequential overview of my work since being accepted as a PhD student in the department, and being awarded an EPSRC grant.

In September 2003, just before joining the university, Dr. Bryson arranged funding for me to attend the 2003 Symposium on Intelligent Motion in Virtual Environments (IMIVE 2003). I found the prospect quite daunting, but it turned out to be an amazing opportunity to meet people and pick up ideas. I talked a lot to the students that attended, finding out about their research and PhD life in general. I also attended all the talks, with those by Fred Brooks, Norm Badler and Daniel Thalmann being most memorable. The most inspirational talk in terms of my current research, however, was given by Electronic Arts' John Buchanan, entitled 'Computer Games: Where Do We Go From Here?' Chatting to him afterwards about my general interests, he recommended I look at the research of John Laird in computer game AI and Mike Gleicher in motion blending. This was my starting point upon joining the university.

For the first month or so I studied these two problems in tandem, looking particularly into Laird's group's work with Soar (Section 4.2). In November 2003, I was given the opportunity to give a talk at the Bath AI Group seminar. This was a completely new experience for me, and will undoubtedly aid my preparation for future conferences. I chose to present the paper by Laird et al. (2002) outlining the Haunt II Project (see Section 3.4.2), as well as giving a brief practical demonstration of Soar.

A few weeks before giving this talk, the BAI seminar was given by Roman Belavkin of Middlesex University. His excellent talk was on the 'entropy of success' but, more importantly for me, the architecture he used for his experiments was ACT-R. Dr. Bryson was also somewhat enthused by ACT-R's potential, and we three discussed the possibility of re-implementing John Mann's ACT-R model of Transitive Inference (TI). This resulted in several months work, culminating in the submission of a paper to CogSci 2004 (see Sections 3.3 and 4.3). My first degree had not required any kind of dissertation of me, so this project was very important for the development of my skills in reading, coding, carrying

out experiments, statistical analysis and writing. It should be noted that Roman Belavkin provided much appreciated help and hints via e-mail throughout the duration of the project.

Since finishing the project, I have concentrated on computer game AI research, as this was what I ‘missed’ the most while doing the cognitive modelling. A more detailed breakdown and justification of this focus can be found in the following section.

## 2 Chosen Area

As I alluded to above, my decision to go in the computer game AI direction, as opposed to animation or cognitive modelling, has come about fairly recently, and thus is not necessarily set in stone. Nevertheless, I will set out and broadly justify my chosen areas of research below.

### 2.1 What?

My primary research topic at present is imitation learning, with a view to applying this to computer game agents. I am more interested in the recognition and adoption of higher-level behaviours (of the kind that can be found in POSH<sup>1</sup> plans (Bryson, 2001)), rather than lower-level (of the kind that are stored in the behaviour libraries referenced by POSH plans). This may enable me to look at the learning of POSH plans, by imitation or other means, and perhaps the more general performance of these plans in computer game environments. Finally, I may look at how imitation learning works in conjunction with other types of planning (HTN, for example), for comparison with POSH.

### 2.2 Why?

Below are some of the questions I want to answer through my research:

- Can the ideas of Demiris and Hayes (1996); Demiris et al. (1997) be applied to virtual agents (ie. in games)?

Demiris has demonstrated imitation learning in robots (see Section 3.4.2). How well can this be transferred to virtual sensors? Thureau et al. (2004) have demonstrated a sort of learning in a virtual environment, but only in the sense of neural network pattern matching, not through high-level behaviour recognition.

- Can POSH plans be learnt?

According to Dr. Bryson, this is still an open research question. The three prerequisites for formulating a POSH plan are finding out **how** to do things, **when** to do them and **what** to do. Can the balance between these be found automatically for emergent behaviours (i.e. new behaviours learnt by the agent)?

- Can computer games be improved by agents who learn from the player?

---

<sup>1</sup>Parallel-Rooted Ordered Slip-Stack Hierarchical

It is my belief that computer game agents could learn through imitation to adapt to the tactics of the individual. Here is one possible process: the first step would be action recognition through virtual sensing, ie. having the ability to know what the human player does in what circumstances. Secondly, an agent would need to be able to classify a given behaviour as successful (leading toward a goal) or unsuccessful (leading away). By adopting successful behaviours and avoiding unsuccessful ones, an agent with similar tactics to the human player is created. So, by using a player's own tactics against him, the agent could learn counter-responses in the same way.

### 3 Literature Review

What follows is a review of all the papers I have read so far. The final section on computer game AI, and in particular the subsection on learning, give the most detail, since this is my chosen topic of research.

#### 3.1 Prior Reading

After making initial enquiries about studying for a PhD at Bath, it was recommended that I read some of Dr. Bryson's papers, picking those that sparked an immediate interest. The first I chose was not a paper as such, but an interim chapter of her PhD thesis (Bryson, 2001) entitled 'How to Make a Monkey Do Something Smart'. The document provided a gentle introduction to her novel architecture and development process, Behaviour Oriented Design (BOD). I then continued to read about BOD and its constituent POSH reactive plans in (Bryson, 1999, 2002, 2003), not to mention the PhD itself. The thesis included a chapter on applying BOD to the Transitive Inference (TI) problem (see Section 3.3).

The paper detailing the LEGO Project (Bryson and Thórisson, 2000) was also important for me, as it introduced me to the animation segmentation problem (Section 3.2) and served as a concrete example of AI used in interactive entertainment (as in computer games, Section 3.4).

The last paper I read prior to arrival was Brooks' seminal 'new AI' paper (Brooks, 1991), just in advance of attending IMIVE 2003. This helped to put BOD in context, and better prepared me for questions about the architecture.

#### 3.2 Animation

What I have referred to as the motion blending problem, is the question of how to smoothly, realistically and (for games in particular) responsively change from one (3D) animated behaviour to another.

As I mentioned in Section 3.1, this problem became apparent when implementing the most complex character in the LEGO Project (Bryson and Thórisson, 2000). Their solution was to build a movement library from scratch which enabled them to manipulate the characters' joints and features directly, rather than attempting to stitch together pre-canned animation sequences. Below, I will look briefly at some comparable approaches.

Snap-Together Motion is a system designed by Gleicher et al. (2003) to transition between motions by blending sequences of motion capture samples.

This is achieved by an animator selecting, with the help of the system, a pose common to a number of motions. Any of these motions can then be blended into any other, by interpolating ‘through’ this pose (with certain constraints enforced to avoid problems such as footskate).

Similar motion sequences, representing basic motions such as ‘walk’ or ‘wave’, are defined by Grünvogel (2003). This system uses ‘clip operators’ to modify specific parameters of a given model, changing the *characteristic* of the motion in question (ie. a ‘happy walk’ becomes a ‘sad walk’). The Verbs and Adverbs system (Rose et al., 1998) is analogous, where the motion models are referred to as ‘verbs’ and the control parameters as ‘adverbs’. The former architecture uses linear blending and specially annotated motions for constraint handling (like Snap-Together Motion) whereas the latter uses radial basis function<sup>2</sup> interpolation and inverse kinematics for constraint handling (Rose et al., 1998, 2001).

A different approach to figure animation is put forward in (Wen et al., 2002): mesh skinning. Rather than a rigid articulated frame (as above) with polygon meshes covering each bone, the figure is represented as a single mesh, and the bones are more logical structures that affect the mesh vertices at joints. Mesh skinning attempts to combat the problem of holes and other skin deformities which are common when using more conventional animation processes.

Methods that might potentially be used to compare and measure the realism of motion produced by the systems above, are outlined by Reitsma and Pollard (2003).

### 3.3 Cognitive Modelling

This field of interest was introduced to me through three key events. Firstly, I came across Dr. Bryson’s TI model whilst reading her PhD (Bryson, 2001) before I started at Bath (see Section 3.1). Secondly, Roman Belavkin presented a good model of non-human behaviour using ACT-R (Belavkin and Ritter, 2003) when he visited Bath (see also Sections 1.3 and 4.3). Thirdly, I was given a project by Mann (2003) attempting to model the TI problem in ACT-R. The model had failed to use the sub-symbolic learning available in ACT-R, meaning a correct ACT-R model of TI still did not exist. Using this background work and (Anderson et al., 2004), I built and tested such a model. Below is a summary of the project write-up by Dr. Bryson and I (Wood et al., 2004).

#### 3.3.1 Summary of the Paper

Transitive inference (TI) formally refers to the process of reasoning whereby one deduces that if, for some quality,  $A > B$  and  $B > C$ , then  $A > C$ . In some domains, such as integers or heights, this property will hold for any  $A$ ,  $B$  or  $C$ , though for others it does not

Harris (1988) showed that primate TI performance on both pairs and triads could be accounted for if we assume that monkeys learn a production rule stack. A *production rule* is a basic AI representation which connects a stimuli to a response. A *stack* is a prioritised list. In the Harris model, each monkey learns one rule per possible stimuli, or up to 5 rules in total. One of two actions is associated with each rule, either *select* or *avoid*. If a subject applies the rule

---

<sup>2</sup>Radial basis functions are functions of the ‘difference’ between two adverbs.

$A \rightarrow s(A)$  (see  $A$  implies select  $A$ ), then it will simply pick up  $A$ , however many other items are present. However, if a subject applies the rule  $A \rightarrow a(A)$  it will pick up anything *but*  $A$ . If more than one other item is present (as in a triad), the subject is at chance for which object it will pick up. If more than two rules could apply, then whichever rule is higher in the stack (has higher *priority*) will be applied.

Although Harris’ hypothesis may seem obscure, it shows a remarkable match to the data. Table 1 shows all of the possible discriminable stacks as identified by Harris and McGonigle (1994).

Table 1: Enumeration of Harris & McGonigle Stacks

#	Rule Depth			#	Rule Depth		
	1	2	3		1	2	3
1	s(A)	s(B)	s(C)	5	a(E)	s(A)	s(B)
2	s(A)	s(B)	a(E)	6	a(E)	s(A)	a(D)
3	s(A)	a(E)	s(B)	7	a(E)	a(D)	s(A)
4	s(A)	a(E)	a(D)	8	a(E)	a(D)	a(C)

The two-tier model hypothesises two learning systems: one for connecting perceptual contexts to actions, and another for prioritising which of those perceptual contexts to attend to if more than one are present simultaneously. A successfully trained two-tier model creates a replication of the production-rule stack model. However, it is dynamic, and as such gives us insight into why animals have trouble learning the initial pairs for the TI task, the sorts of mistakes they may make, and the impact of training régimes. The first tier of the model is a single-vector neural network (NN) which learns the prioritisation of the stimuli, the second tier is a set of small two-item vectors which each learn to associate a rule with one of the stimuli. For further details see (Bryson and Leong, 2004).

As for the above models, ACT-R also learns production rules, but any number of these rules may have their preconditions for firing satisfied at any given time. In this case, ACT-R’s conflict-resolution system selects the rule with highest *utility* value. Rule utilities are changed by ACT-R’s sub-symbolic processing system. The stack model proposed by Harris and McGonigle (1994) attempts to fit the McGonigle and Chalmers (1977) triad data to any of the eight discriminable correct stacks (Table 1). In contrast, the ACT-R agents do not learn a totally ordered stack, but effectively a pair of stacks. The two possible pairs are:

Hybrid Stack 1    s(A)a(E)s(B) & a(E)s(A)s(B)  
Hybrid Stack 2    s(A)a(E)a(D) & a(E)s(A)a(D)

For three of the five individual test subjects for whom McGonigle and Chalmers (1977) triadic data is available, one of the hybrid stacks fits better than any of the eight others Harris proposes (Table 2). This implies that some monkeys do not form a total ordering, so their choices cannot be perfectly modelled by a simple production-rule system.

For Bump and Brown, however, our model is rejected ( $p < 0.01$ ), suggesting that other monkeys do come up with a total ordering, which cannot be well

Table 2: Individual Analysis of 1977 Triadic Choice Data

	A	B	C	D	E	$\chi^2$	$p(O)$
Bill	55	17	20	8	0	-	-
HS2	52.5	17.5	17.5	12.5	0	2.1	n.s.
S 4	60	15	15	10	0	2.8	n.s.
Blue	55	25	14	6	0	-	-
HS1	52.5	22.5	12.5	12.5	0	4.0	n.s.
S 3	60	20	10	10	0	4.9	n.s.
Bump	53	34	8	4	1	-	-
HS1	52.5	22.5	12.5	12.5	0	13.3	< 0.01
S 2	60	30	5	5	0	3.4	n.s.
Brown	36	29	24	11	0	-	-
HS2	52.5	17.5	17.5	12.5	0	15.3	< 0.01
S 7	35	25	25	15	0	1.8	n.s.
Roger	51	26	6	17	0	-	-
HS1	52.5	22.5	12.5	12.5	0	5.6	n.s.
S 5	45	25	15	15	0	6.5	< 0.1

modelled in ACT-R. The two-tier model does support both models, although its admittedly simplistic learning rule tends to favour the total ordering. These results suggest that an improved priority-learning rule for either the two-tier model or ACT-R could result in a highly accurate model of TI and probably task learning in general. Integrating production compilation (Anderson et al., 2004) and dynamic noise (Belavkin and Ritter, 2003) into the ACT-R models could form two important stages in this modification, and initial work in this direction has shown promise.

### 3.4 Computer Game AI

There are now a number of flourishing research groups, conferences and journals for this subject; reviewed below are the articles I found most helpful in gaining a better perspective on its open research issues.

#### 3.4.1 High-Level

The computer games industry is the most profitable entertainment trade in the world, including even the film industry. It remained unaffected by the recent downturn in the IT market. In addition, it provides a huge test ground for situated cognition and motivation for researchers to pursue human-level AI (i.e. machines that can pass the Turing Test).

A good overview of the potential applications of AI to computer games is given by Laird and van Lent (2001). In their attempt to motivate academic research in this domain, they describe games as the ‘killer’ application of human-level AI. They explain how other research domains encourage specialisation, whereas games require agents to be fully integrated with a broad variety of AI skills. Most of the paper is taken up with describing the different computer game genres and the roles of AI agents within them; similar in nature to section 3.4.2

below. A more detailed breakdown of the kinds of skills needed by computer game agents, and their effect on limited resources such as memory and the CPU, can be found in (Laird, 2000).

Cass (2002) comments upon the relationships and inherent differences between academic and industrial AI research. He claims that computer game developers, bound by stricter deadlines and budgets, will tend to stick with tried and tested AI programming methods (namely finite-state machines (FSMs) and scripting). Conversely, the academic community is more willing to trade stability and consistent playability for novelty and biological plausibility.

Both Laird and Cass agree that due to recent hardware advances, the graphics race<sup>3</sup> has all but run its course and consequently, AI is becoming the new metric for computer game quality.

### 3.4.2 Different Game Genres and Their Needs

This is by no means an exhaustive list of computer game genres; I have focused on those more common in the literature and most pertinent to my research interests. Many of the newest games are in fact hybrids of several genres.

#### First Person Shooters

A First Person Shooter (or FPS) is an action-oriented game where the player assumes the viewpoint of a single character that they control within a 3D virtual environment. Typically, the gameplay involves moving around a maze-like complex, killing enemies (AI- or human-controlled avatars) and collecting ‘power-ups’ (such as health bonuses or extra ammunition). More advanced FPSs may present scenarios with higher objectives, such as Capture The Flag, which require team play. Examples of this genre include Unreal Tournament 2003, Halo and Half-Life.

John Laird’s research group at the University of Michigan has worked extensively with this type of game. In (Laird, 2001b), he describes their development of ‘Quakebots’; avatars controlled by the rule-based Soar engine (see section 4.2) that play the FPS Quake II. As well as being endowed with basic tactics, such as which gun to use where and how, Laird’s Quakebots utilise *anticipation*, given a reasonable amount of data regarding an opponent’s position and motion, and appropriate low-risk circumstances. By applying their own pre-programmed rules internally, the agents are able to ascertain what they would do in the state of their opponent, thereby approximately anticipating enemy actions. This prediction could in turn be used, for example, to set an ambush or steal a much needed power-up. This process is more fully described in (Laird, 2001a).

Other research in this area includes that of van Waveren and Rothkrantz (2002), who have constructed an artificial player for Quake III. They use a multi-layered, modular architecture where the central ‘AI network’ is essentially an FSM. Bot navigation is handled by their so-called Area Awareness System, which uses an algorithmically-generated 3D Binary Space Partitioning tree, a significant improvement upon the standard human-generated 2D waypoints system of many FPSs.

---

<sup>3</sup>Graphics has been the chief selling point for most computer games to date.

## Adventure Games

Adventure Games differ from FPSs in that the emphasis is on information-gathering and puzzle-solving rather than real-time combat. Interactive Fiction games are the oldest sub-class of adventure games<sup>4</sup>, requiring textual commands to be input by the player, and providing on-screen, (primarily) textual output. Then came ‘point ‘n’ click’ games, with a 2D GUI for instructing the protagonist and selecting from lists of presented options (e.g. *The Secret of Monkey Island*). Many of the latest adventure games are fully 3D virtual environments; some played from the point of view of the controlled character as in FPSs (eg. *CSI: Crime Scene Investigation*), and some with each ‘room’ having a fixed camera position (eg. *Broken Sword: The Sleeping Dragon*).

In a paper outlining his work on Quakebots, Laird points out the limitations of FPSs as a platform for rich AI characters. After all they are “essentially computerised punching bags” (Laird, 2002). Laird et al. (2002) describe plans and initial work done by his group on an adventure game called *Haunt II*. It is based upon the Unreal Tournament (UT) engine which, although designed for FPS games, has an accessible scripting language incorporating abilities such as sensing and pathfinding also useful in adventure games. By augmenting the UT engine with extra senses and state, and coupling it to a Soar engine running the AI, they proposed creating an adventure game whose story is driven by the interaction of the player-character with the AI-controlled ‘supporting cast’. Each non-player character (NPC) would have its own Soar agent dictating its actions.

An extension of this work is proposed in (Magerko, 2002). He observes that if the NPCs were granted total autonomy, advancing the plot of the game to satisfy given narrative prerequisites becomes very difficult if the characters are to remain complex and believable. The solution he delineates is the Interactive Drama Architecture (IDA), which includes a director agent. This director is provided with the writer’s plot specifications, the state of the game world and the actions of the player, and can ‘direct’ the NPCs, encouraging them to take actions that should advance the storyline. (Magerko and Laird, 2003) describes a first implementation of the IDA in the *Haunt II* environment (see above). The paper also details the representation of the story as a graph of partially-ordered plot events, and the necessity of user modelling (see below) to determine the most likely plot path. (Magerko and Laird, 2002) discusses the architecture’s potential application to adaptive military training simulators.

The work of Cavazza et al. (2003) is a fascinating example of interactive storytelling where image (particularly gesture) recognition techniques are used to embed the user into a virtual scene, rather than being represented by a distinct avatar. It also serves as an example implementation of Hierarchical Task Network (HTN) Planning in AI (see below).

## God Sims

The role of the player in a God Sim is very different from that in the genres described above. Rather than being immersed in the game environment, the

---

<sup>4</sup>Advent is generally considered to be the first IF game, which dates to c. 1975.

player is an observer with the ability to alter the world and/or its residents in some way. Often the game itself defines no explicit levels or goals; these are dependent upon the whims of the individual player. Two common categories of God Sims are:

- Those where the player constructs a world to serve its population<sup>5</sup> at a macro level (eg. SimCity 4, Sim Theme Park), and
- Those where the player manipulates the individual characters and/or their surroundings on a micro level (eg. The Sims, Black and White).

Charles et al. (2002) identify a similar conflict between interaction and narrative as Magerko (2002) (see above). However, since they are working in the God Sim domain, the game’s entertainment is derived from the autonomy of and interaction between the NPCs. Their solution, then, is to limit how the *player* can interact with the environment, namely through manipulating objects or passing information directly to the NPCs. Each NPC uses HTN planning (see below) to make decisions and execute actions within the environment. The variance in storylines comes about by introduced randomness, plan interference and, of course, user intervention.

Engaging, autonomous NPC interaction is also the goal of the Proactive Persistent Agent (PPA) architecture project (Mac Namee et al., 2003; Mac Namee and Cunningham, 2003). They claim that in other games and architectures, the virtual actors seem to have no ‘life’ outside of the sphere of their dealings with the human player. This detracts from the believability of the player’s observations and is something they wish to remedy in their work (hence *Persistent* in the project title). Rather than the HTNs of the Charles et al. (2002) system, the actors here are controlled using Fuzzy Cognitive Maps (FCMs), described by Mac Namee et al. (2003) as:

“...directed graphs in which nodes represent fuzzy concepts and arcs represent fuzzy rules, or the causal flow between concepts.”

A key component of the PPA architecture is the  $\mu$ -SIC system; the social unit for determining PPA interaction. The system uses an Artificial Neural Network (ANN) with three sets of inputs: those for ‘personality’, ‘mood’, and ‘relationship’ to the agent in question. The outputs map to the various possible interactions the PPA could carry out with the other agent.

Due to the overlapping requirements of God Sims and Adventure Games in the area of autonomous virtual actors, the principles and techniques described in the above two sections could be applicable in both game domains.

### Real-Time Strategy Games

Real-Time Strategy (RTS) Games, like God Sims, typically present a more topographical player perspective. The play, however, is more like a board game, where the NPCs are simpler, generic pawns in a battle between higher powers. Common in-game tasks include collecting resources, building bases (and defending them from assault) and mustering forces (and preparing them for missions).

---

<sup>5</sup>Of course, the player may prefer to design a world specifically to antagonise its population!

Goals range from specific tasks such as protecting an individual agent or sneaking into and enemy base to simply surviving while you wipe out your enemies from the map. Warcraft III, Command & Conquer: Generals and Medieval: Total War are examples from this genre. The role of AI in RTS Games is two-fold. Low-level AI is needed to control individual units (eg. path-planning), and high-level AI is needed to form strategies for and command entire armies.

In contrast with FPSs like Unreal Tournament, commercial RTS Game developers have been thus far reluctant to make their APIs publicly available. This poses an obvious problem for researchers seeking to test their AI in a worthy environment; a problem which Buro (2002) is attempting to solve with the ORTS (Open RTS) Project. As suggested by the project title, the work of his research group at the University of Alberta is open source, and collaboration is encouraged. In fact, as did Laird and van Lent (2001) for computer games in general, Buro and Furtak (2004) spend some time explaining the various AI research areas relevant to RTS Games.

One of the system's main selling points is that it is 'hack-free'. In other words, the full game simulation is run on a secure server and a given client is passed only the game data that should be accessible to that client. In contrast, most commercial releases run simulations on the client side (to improve network performance), leaving data masking up to the client and thus making it possible to use 'map-revealing' hacks and other cheats. The project is still in its developmental stages<sup>6</sup>, but they have already implemented kinematics (including basic collision detection) and vision systems.

### 3.4.3 Improving NPCs

As is clear from the above reviews, improving NPCs is an underlying theme in much game AI research. In this section I will look over some techniques that have contributed to progress made in this area.

#### Planning

One technique is to add planning modules to agents, allowing them to concatenate strings of primitive actions into more complex behaviours. I have already mentioned Quakebots (Laird, 2001a) as an example of using planning to improve NPCs. Here the anticipated actions of the human player are used to formulate counteractive plans. His improvements are partly tied in to the way Soar makes decisions and plans (see Section 4.2).

More generally, HTN planning (Nau et al., 1998) has also been applied to this end (Charles et al., 2002; Cavazza et al., 2003; Hawes, 2003). Here, complex, high-level plans are recursively decomposed into simpler constituent tasks by methods held in the planning system's knowledge base. In many applications, the plan would need to be decomposed into primitive actions to be executable, but Hawes (2003) describes a way of utilising partially-solved HTN plans as part his novel anytime algorithm (A-UMCP). In his PhD, he reviews the applicability of several different planning approaches to game worlds, comparing fast (eg. GraphPlan), reactive (eg. PRS) and continual (eg. CPEF) planners.

---

<sup>6</sup>at time of writing

## Learning

The second technique for NPC improvement is learning. In this context, planning and learning are complementary. Plans can be learnt to increase efficiency, and learning boosts domain knowledge and can improve plan quality.

Yet again, the Quakebots (Laird, 2001a) serve as good examples of this. Once a counterplan has been generated, executed and deemed as successful, they use Soar's chunking mechanism (Section 4.2) to learn it. The next time these circumstances arise, they can execute the tested plan immediately, instead of wasting time reformulating.

The above is an implementation of online learning, but some offline learning methods have shown similar promise:

- Spronck et al. (2002) used evolutionary algorithms to evolve an enemy spaceship with superior duelling skills for the strategy game PICOVERSE. They also unexpectedly found that they could identify weaknesses in the standard, scripted ships' tactics by analysing the tactics of the evolved ship. Although the domain was strategy, only combat behaviours were produced, and nothing higher-level.
- Thureau et al. (2004) came at the problem from a pattern matching perspective. The FPS Quake II allows whole games to be recorded, with many of these so-called demos available on the Internet. By training neural networks with these game state data, they were able to produce coherent movement and aiming behaviours in their bots. This is also an example of imitation learning, since the input behaviours were those of human players.

A more biologically-inspired model of imitation learning is presented by Demiris and Hayes (1996), in the field of robotics. They begin by recounting various theories of supposed natural imitation learning, culminating in Meltzoff's Active Intermodal Matching mechanism:

“When infants see an act they use proprioceptive information from their own body movements to iteratively match the visual information they perceive.”

Their architecture follows this principle; data from a robot's internal and external sensors are combined to calculate the deviation of the robot's position from the observed human expert's position, in real-time. Acting to minimise this deviation corresponds to imitation. At first, the robot knows only how to copy its teacher, but can learn 'good' actions to take by associating environmental conditions with successfully imitated actions. An advantage of this type of learning is that the expert need not explicitly teach or even communicate with the learner: remote observation is sufficient (Demiris and Hayes, 1996; Demiris et al., 1997).

## User Modelling

The third and final technique I will discuss here is user modelling. A paper generally motivating research in this area, and particularly assessing the applicability of recent advances in intelligent tutoring and education systems to

interactive entertainment, is given by Beal et al. (2002). They concede that game environments are ordinarily far less constrained and well understood than pedagogical tools, but suggest that some of the same modelling mechanisms (ie. machine learning) could be well integrated, principally with a view to appealing to a wider gaming audience.

A more concrete example of user modelling is goal recognition: attempting to discern the user's current goal from their state and recent actions. Magerko and Laird (2003) use it in their IDA (see above) to predict the likelihood of the user satisfying the plot requisites of a given scene. The director agent can then take action to try to alter the user's goal, if appropriate. Albrecht et al. (1998) applied a Dynamic Belief Network (DBN) to sets of recorded user actions from the Multi-User Dungeon (MUD) 'Shattered Dreams'. Their aim was accurate *quest* prediction given preceding *locations* and *actions*, these three primitives making up the nodes in the DBN. They met with some success, improved by the classification and filtration of certain 'noise' words.

## 4 Systems Review

Over the past six months, I have investigated a number of different software packages. The most interesting and useful of these are described below.

### 4.1 Blender

Blender is a 3D design suite, with tools for 3D mesh modelling, animation (including articulated figures), rendering and post-production. I came across it when I was looking for an environment I could potentially use for working on the motion blending problem (see Section 3.2). The Blender 2.0 Manual contains a Quickstart section which covers all the basic features, and having worked through that I looked at some of the more relevant sections in detail.

The two most impressive components of Blender are:

1. The ability to bind articulated bone structures to parts of a 3D mesh surface. It is possible to place the joints within the structure and then associate groups of vertices with each. By moving the bones, the associated 'skin' moves in tandem, although not free from the sorts of problems described in Wen et al. (2002) (see Section 3.2).
2. The ability to precisely automate animation via Blender's Python Scripting Engine. Necessary for interfacing the required complex motion models with the 3D mesh models. Could also be used to imbue the environment with 'physics'.

These features make up for the complexity of the system in general and the GUI in particular. I found Blender preferable to Maverik, and certainly to direct graphical manipulation using Open-GL which I also experimented with. If I were to re-examine this problem at a later date, this is where I would return.

### 4.2 Soar

On the Soar homepage, the architecture is described as:

‘... a general cognitive architecture for developing systems that exhibit intelligent behaviour.’

It is the baby of Allen Newell and John Laird, the latter of which continues its development, based at the University of Michigan. Having been pointed in Laird’s direction by John Buchanan (see Section 1.3), I found that much of his games research to date made use of Soar to drive Quakebots and other NPCs (Section 3.4.2). Below is a breakdown of the mechanisms used by Soar:

**Objects** with **attributes** and **values** encode all temporary knowledge within the system, ie. all states that can be tested.

**Productions** encode all permanent knowledge within the system. These take the form of IF-THEN rules which test and alter the states of objects via **operators**. All operators are given a **preference** (either comparative or numerical) which determines which are executed in the event of several productions being satisfied simultaneously.

**Problem spaces** are state spaces defined by the particular set of operators that are relevant to a given task or subtask, ie. the locations defined by move operators.

**Automatic subgoaling** is used to generate new goals if, for example, a higher-level goal needs to be subdivided or reaches a state of **impasse**, where no operators are applicable.

**Chunking** is the learning mechanism. Here, Soar can generate new productions (ie. permanent knowledge) by associating states with actions which brought it ‘nearer’ to a given goal. These are found by trial and error.

The Soar 8 Tutorial (Laird, 2003) provides a practical tour through these mechanisms, with the help of example Soar game agents in Eaters (a Pac-Man-like maze game) and TankSoar (similar, but with more advanced sensors, weapons and power-ups).

As part of my exploration of the applications of Soar in games, I gave a talk to Bath AI Group (BAI) based on (Laird et al., 2002) (see Section 3.4.2). As I explained then, Soar has a strange idiosyncrasy which is a product of the way knowledge is encoded. Since knowledge about object states can only be stored as temporary knowledge (ie. in working memory), in order to *initialise* object states they must be encoded in production operators that are immediately satisfied and never retracted. This is somewhat counter-intuitive and is not the case in ACT-R (see Section 4.3).

### 4.3 ACT-R

I was made aware of the basic principles of ACT-R by a review given in (Bryson, 2001), and a few cursory discussions about the work of Mann (2003). It was not until a seminar given by visiting speaker Roman Belavkin, however, that I looked into the architecture for myself in more detail.

### 4.3.1 What Is ACT-R?

The following description is taken from the ACT-R homepage:

“ACT-R is a cognitive architecture: a theory for simulating and understanding human cognition.”

At the highest level, the architecture comprises of a collection of buffers storing internal and external state, monitored and changed by production rules. Anderson et al. (2004) detail the main components of the system; below is a summary:

**The Procedural Memory Module** is ACT-R’s decision-making system. This module contains productions in the form of IF-THEN rules (like Soar). The IF part of the rule can check various buffer states (see below) and, if satisfied, the rule becomes a candidate to fire. The THEN part alters buffer states, and the decision cycle begins again.

**The Declarative Memory Module** deals with ACT-R’s factual processing. Units of declarative memory are called **chunks**, and are used to represent all the facts known by the system. Productions can request that chunks be transferred from declarative memory into the **retrieval buffer**, where they can then be matched by other productions.

**The Perceptual-Motor System** enables ACT-R to interact with its environment. The perceptual buffers (ie. visual, auditory) change state with the world, and their contents can be matched by productions. Conversely, the contents of the motor buffers (ie. manual, vocal) can be altered by productions and cause ACT-R to act upon the world.

**The Goal Module** keeps track of ACT-R’s current goal, and the progress that is being made toward it. The latter happens in the **goal buffer**, which can be both matched and updated via productions.

So far I have only covered the symbolic attributes of ACT-R; a lot of its power comes from sub-symbolic processing capabilities. The speed and success of chunk retrieval is governed by **latency** and **activation**, representing how ‘strong’ a given memory is. For action selection, the highest rule **utility** decides which candidate production is fired. Utilities are incremented and decremented depending upon the relative success or failure respectively of a given rule in helping to achieve a given goal. ACT-R is also capable of **compiling** (learning) new productions using a similar method to Soar (see Section 4.2). Since ACT-R is specifically a model of human cognition, the many available parameters default to approximate human reactions and response times. Both being production-based cognitive architectures, Soar and ACT-R are similar in many ways. Soar is perhaps more generally applicable, but I found ACT-R more intuitive, more powerful and easier to program.

### 4.3.2 Modelling TI in ACT-R

Section 3.3.1 summarises the results of the paper Dr. Bryson and I submitted to CogSci 2004. Here I will give a brief overview of my experience of writing the ACT-R models for this project.

Firstly, I worked through the tutorials found on the ACT-R web site. Each chapter concluded with a challenge exercise to build a model to approximate real human data within acceptable tolerances; an invaluable skill for the forthcoming project. I then examined John Mann's code to see what I could glean from his attempt at solving the same problem. It turned out he had not used the sub-symbolic system, but had stored his own statistical data outside of ACT-R. Because of this, I had to start a new model from scratch, although his code did help me to understand how to build a system to interact with an ACT-R model.

There are two parts to every ACT-R project; the first is writing a system which presents stimuli to the model, and the second is writing the model itself. In reality, the development is iterative, with progress on one part prompting progress on the other. The first part requires an understanding of Lisp, the language ACT-R is built on, and one that I had no previous experience in. (Steele Jr., 1990) was a very helpful reference for this.

In the end I wrote two main models<sup>7</sup>. The first could immediately choose to *select* or *avoid* any stimulus, depending on rule utilities. The second had to *focus* on a stimulus, and then *select* or *avoid* it. I also wrote a number of sub-models, making use of dynamic noise (Belavkin and Ritter, 2003) and production compilation, but these were never fully validated or included in the write-up.

## 4.4 Latest Experiments

Here are a few of my more recent discoveries, found in conjunction with my search for a suitable game development environment.

### 4.4.1 Breve

Breve is an environment for modelling 3D and Artificial Life simulations. Coding is done in *steve*, an object-oriented language based on C. Many useful *steve* classes are included with the *breve* installation, such as those representing 3D solids, joints for connecting multi-body objects, planes, 2D grids, cameras and lights. *Breve* also gives the option to enable physics in the environment, but the documentation warns that this can be processor-intensive and should only be used for bona fide 3D simulations. Simpler physics should be coded separately on a case-by-case basis.

Having read the documentation and briefly experimented with the software, I believe *breve* could be used to create simple games for my research purposes. Advantages of using an A-Life simulator include having full creative control over the in-game objects, and having the ability to advance the simulation frame-by-frame, recording the world state at each iteration. The main disadvantage I can see is development time: having to come up with a concept for a game which demonstrates my work, implementing it from scratch and working out what kind of environmental physics (if any) would be necessary.

### 4.4.2 Robocode

Robocode is actually the name of an Internet AI programming competition, but also lends its name to the software used to create and test the 'robots' that

---

<sup>7</sup>Referred to as *ACT-R-1* and *ACT-R-2* in the paper.

compete. The setting for the competitors is a rectangular 2D arena where AI-controlled tanks (or teams of tanks) do battle to destruction (somewhat similar to TankSoar - see Section 4.2). The code is basically a Java package, with class hierarchies for the robot and its various sensors and weapons, and a couple of nice GUIs for watching and governing the action.

Although learning violent behaviours is perhaps not an ideal starting point for my research, Robocode is appealing for other reasons. A lot of code for basic perception and action is provided, the environment is relatively simple (compared to Unreal Tournament, for example) and it is essentially a game already. Whether I will have the freedom to create robots which can exhibit and learn rich enough behaviours to be useful is a key question which I hope to answer through further experimentation.

#### 4.4.3 Other Possibilities

At present, I am far from making a final decision about which game environment to use. Other possibilities I have looked at include Mason, which is a Java-based A-Life simulator similar to breve, and Unreal Tournament (UT), which has been used extensively for game AI research in the past (see Section 3.4). Both Soar and ACT-R have been interfaced with UT via its API, so there may be some scope there for comparing systems I have experience with. I intend to continue my search, starting with some contacts I have recently acquired.

## 5 Plans

The purpose of this final expository section of my report is to clearly set out my work targets as they stand at present.

### 5.1 Next Six Months

My research goals for the next six months of my PhD are summarised below:

- To read further into the machine learning literature, specifically with respect to imitation learning. It is important to find the domains this has been applied to, how and to what extent it has been applied, whether any techniques or architectures could be adapted to computer game agents, and if anyone has done this particular sort of work before.
- To settle upon a development environment for my agents (see Section 4.4). Initially I need just to set up a few trivial scenarios to demonstrate that my ideas can work in practise. If successful, I will then look to more complex (probably commercial) game engines.
- To implement some simple example agents in my chosen environment. The first task here is to set up the code constructs necessary for basic action recognition, plan compilation, memory, etc. in a trivial scenario.
- (Optional) To look into the theoretical complexity of POSH reactive plans. Nothing has been published on this to date, and it may make a good journal article or conference paper.

## 5.2 Rest Of PhD

Clearly, without any foreknowledge regarding the success of the above research plan, I can only talk about the rest of my PhD in the most general terms.

One consideration is how important a feature I want biological plausibility to be in my research. The advantages of keeping it central are that it utilises the experience in cognitive modelling I gained from the CogSci project, and it would probably make my research appealing to a wider audience. Disadvantages include the fact that if I focus on biological plausibility, I am likely to have to sacrifice some system speed and stability. This could damage my application domain of computer games, since reliability is so important to the games industry (Cass, 2002). In practise, the models I build are likely to be modular, and as long as I have one or other of these goals in mind for each module, I will have something worthwhile to write about.

Another consideration arises from my initial desire to study robots. Should I manage to find a new technique for imitation learning in virtual environments, could it cross *back* over into the robotics domain? There is no way I could comment at this stage, but I suppose it hinges upon the generality of the theory.

## 6 Summary

I believe this report combines a thorough and comprehensive review of the work I have done so far with a clear statement of my current research intentions. I found writing it admittedly laborious, but also useful for collating my thoughts and documenting things I may need to reference in the future which would otherwise have gone undocumented. As I stated in the introduction, I am hoping that at least some of this report may be adaptable for inclusion in my thesis. Finally, I would like to thank Dr. Joanna Bryson for her feedback with regards to the structure and content.

## References

- (1990). The secret of monkey island. Lucasfilm Games.
- Albrecht, D. W., Nicholson, A. E., and Zukerman, I. (1998). Knowledge acquisition for goal prediction in a multi-user adventure game. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 1–12.
- Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., and Qin, Y. (2004). An integrated theory of mind.
- Beal, C. R., Beck, J., Westbrook, D., Atkin, M., and Cohen, P. R. (2002). Intelligent modeling of the user in interactive entertainment. In *AAAI Stanford Spring Symposium*, Stanford, CA.
- Belavkin, R. V. and Ritter, F. E. (2003). The use of entropy for analysis and control of cognitive models. In Detje, F., Dörner, D., and Schaub, H., editors, *Proceedings of the Fifth International Conference on Cognitive Modelling*, pages 21–26, Bamberg, Germany. Universitäts-Verlag Bamberg.

- Brooks, R. A. (1991). Intelligence without representation. *Artificial Intelligence*, 47:139–159.
- Bryson, J. J. (1999). Hierarchy and sequence vs. full parallelism in action selection. In Ballin, D., editor, *Intelligent Virtual Agents 2*, pages 113–125, Salford, UK.
- Bryson, J. J. (2001). *Intelligence by Design: Principles of Modularity and Coordination for Engineering Complex Adaptive Agents*. PhD thesis, MIT, Department of EECS, Cambridge, MA. AI Technical Report 2001-003.
- Bryson, J. J. (2002). Modularity and specialized learning: Reexamining behavior-based artificial intelligence. In Butz, M. V., Gérard, P., and Sigaud, O., editors, *Adaptive Behavior in Anticipatory Learning Systems*, Edinburgh.
- Bryson, J. J. (2003). The behavior-oriented design of modular agent intelligence. In Kowalszyk, R., Müller, J. P., Tianfield, H., and Unland, R., editors, *Agent Technologies, Infrastructures, Tools, and Applications for e-Services*, pages 61–76. Springer.
- Bryson, J. J. and Leong, J. C. S. (2004). A two-tiered model of primate transitive performance: Separate memory systems for rule-stimulus association pairs and their prioritisations. in preparation.
- Bryson, J. J. and Thórisson, K. R. (2000). Dragons, bats & evil knights: A three-layer design approach to character based creative play. *Virtual Reality*, 5(2):57–71.
- Buro, M. (2002). ORTS: A hack-free RTS game environment. In *Proceedings of the International Computers and Games Conference 2002*, Edmonton, Canada.
- Buro, M. and Furtak, T. (2004). RTS games as test-bed for real-time research. Submitted to Information Sciences.
- Cass, S. (2002). Mind games. *IEEE Spectrum*, pages 40–44.
- Cavazza, M., Martin, O., Charles, F., Mead, S. J., and Marichal, X. (2003). Users acting in mixed reality interactive storytelling. In *2nd International Conference on Virtual Storytelling*.
- Charles, F., Mead, S. J., and Cavazza, M. (2002). Generating dynamic storylines through characters’ interactions. *International Journal of Intelligent Games and Simulations*, 1(1).
- Demiris, J. and Hayes, G. M. (1996). Imitative learning mechanisms in robots and humans. In *Proceedings of the 5th European Workshop on Learning Robots*, pages 9–16, Bari, Italy.
- Demiris, J., Rougeaux, S., Hayes, G. M., Berthouze, L., and Kuniyoshi, Y. (1997). Deferred imitation of human head movements by an active stereo vision head. In *Proceedings of the 6th IEEE International Workshop on Robot Human Communication*, pages 88–93, Sendai, Japan. IEEE Press.

- Gleicher, M., Shin, H. J., Kovar, L., and Jepsen, A. (2003). Snap together motion: Assembling run-time animation. In *2003 Symposium on Interactive 3D Graphics*.
- Grünvogel, S. M. (2003). Dynamic character animation. *International Journal of Intelligent Games and Simulations*, 2(1).
- Harris, M. R. (1988). *Computational Modelling of Transitive Inference: a Micro Analysis of a Simple Form of Reasoning*. PhD thesis, University of Edinburgh.
- Harris, M. R. and McGonigle, B. O. (1994). A model of transitive choice. *The Quarterly Journal of Experimental Psychology*, 47B(3):319–348.
- Hawes, N. A. (2003). *Anytime Deliberation for Computer Game Agents*. PhD thesis, School of Computer Science, The University of Birmingham, Birmingham, B15 2TT.
- Laird, J. E. (2000). Design goals for autonomous synthetic characters.
- Laird, J. E. (2001a). It knows what you’re going to do: adding anticipation to a Quakebot. In Müller, J. P., Andre, E., Sen, S., and Frasson, C., editors, *Proceedings of the Fifth International Conference on Autonomous Agents*, pages 385–392, Montreal, Canada. ACM Press.
- Laird, J. E. (2001b). Using a computer game to develop advanced AI. *Computer*, 34(7):70–75.
- Laird, J. E. (2002). Research in human-level AI using computer games. *Communications of the ACM*.
- Laird, J. E. (2003). The Soar 8 tutorial. University of Michigan.
- Laird, J. E., Assanie, M., Bachelor, B., Benninghoff, N., Enam, S., Jones, B., Kerfoot, A., Lauver, C., Magerko, B., Sheiman, J., Stokes, D., and Wallace, S. (2002). A test bed for developing intelligent synthetic characters. In *AAAI Spring Symposium on AI and Interactive Entertainment*.
- Laird, J. E. and van Lent, M. (2001). Human-level AI’s killer application: Interactive computer games. *AI Magazine*, 22(2):15–25.
- Mac Namee, B. and Cunningham, P. (2003). Creating socially interactive non player characters: The  $\mu$ -SIC system. *International Journal of Intelligent Games and Simulations*, 2(1).
- Mac Namee, B., Dobbyn, S., Cunningham, P., and O’Sullivan, C. (2003). Simulating virtual humans across diverse situations. In *Proceedings of Intelligent Virtual Agents ’03*, pages 159–163.
- Magerko, B. (2002). A proposal for an interactive drama architecture. In *AAAI 2002 Spring Symposium Series: Artificial Intelligence and Interactive Entertainment*.
- Magerko, B. and Laird, J. E. (2002). Towards building an interactive, scenario-based training simulator. In *10th Computer Generated Forces and Behavior Representation Conference*, Orlando, FL.

- Magerko, B. and Laird, J. E. (2003). Building an interactive drama architecture. In *1st International Conference on Technologies for Interactive Digital Storytelling and Entertainment*, Darmstadt, Germany.
- Mann, J. E. (2003). Comparison between using ACT-R and behaviour oriented design methodology to model the learning of transitive inference. Final Year Project, University of Bath.
- McGonigle, B. O. and Chalmers, M. (1977). Are monkeys logical? *Nature*, 267:694–696.
- Nau, D. S., Smith, S. J. J., and Erol, K. (1998). Control strategies in HTN planning: Theory versus practice. In *AAAI/IAAI Proceedings*, pages 1127–1133.
- Reitsma, P. S. A. and Pollard, N. S. (2003). Perceptual metrics for character animation: Sensitivity to errors in ballistic motion. In *ACM Transactions on Graphics, SIGGRAPH 2003 Proceedings*, volume 22(3), pages 537–542.
- Rose, C., Cohen, M. F., and Bodenheimer, B. (1998). Verbs and adverbs: Multidimensional motion interpolation. *IEEE Computer Graphics and Applications*, 18(5):32–40.
- Rose, C., Sloan, P., and Cohen, M. F. (2001). Artist-directed inverse-kinematics using radial basis function interpolation. In *Proceedings of EUROGRAPHICS 2001*, volume 20(3).
- Spronck, P., Sprinkhuizen-Kuyper, I., and Postma, E. (2002). Evolving improved opponent intelligence. In Medhi, Q., Gough, N., and Cavazza, M., editors, *GAME-ON 2002 3rd International Conference on Intelligent Games and Simulation*, pages 94–98. SCS Europe Bvba.
- Steele Jr., G. L. (1990). *Common Lisp: The Language*. Digital Press, Bedford, MA, second edition.
- Thureau, C., Bauckhage, C., and Sagerer, G. (2004). Learning human-like movement behavior for computer games. In *Proceedings of the 8th International Conference on the Simulation of Adaptive Behavior (SAB'04)*. To appear.
- van Waveren, J. and Rothkrantz, L. (2002). Artificial player for Quake III arena. *International Journal of Intelligent Games and Simulations*, 1(1).
- Wen, Z., Mehdi, Q., and Gough, N. E. (2002). Animating intelligent games characters using mesh skinning. *International Journal of Intelligent Games and Simulations*, 1(1).
- Wood, M. A., Leong, J. C. S., and Bryson, J. J. (2004). ACT-R is almost a model of primate task learning: Experiments in modelling transitive inference. Submitted to CogSci 2004.