# COMBINED ARTIFICIAL INTELLIGENCE BEHAVIOUR SYSTEMS IN SERIOUS GAMING

MPhil Thesis

David Irvine

Institute of Creative Technologies

De Montfort University

This thesis is submitted in partial fulfilment of the requirements for the Master of Philosophy.

# ABSTRACT

This thesis proposes a novel methodology for creating Artificial Agents with semi-realistic behaviour, with such behaviour defined as overcoming common limitations of mainstream behaviour systems; rapidly switching between actions, ignoring "obvious" event priorities, etc. Behaviour in these Agents is not fully realistic as some limitations remain; Agents have a "perfect" knowledge about the surrounding environment, and an inability to transfer knowledge to other Agents (no communication).

The novel methodology is achieved by hybridising existing Artificial Intelligence (AI) behaviour systems. In most artificial agents (Agents) behaviour is created using a single behaviour system, whereas this work combines several systems in a novel way to overcome the limitations of each. A further proposal is the separation of behavioural concerns into behaviour systems that are best suited to their needs, as well as describing a biologically inspired memory system that further aids in the production of semi-realistic behaviour.

Current behaviour systems are often inherently limited, and in this work it is shown that by combining systems that are complementary to each other, these limitations can be overcome without the need for a workaround. This work examines in detail Belief Desire Intention systems, as well as Finite State Machines and explores how these methodologies can complement each other when combined appropriately. By combining these systems together a hybrid system is proposed that is both fast to react and simple to maintain by separating behaviours into fast-reaction (instinctual) and slow-reaction (behavioural) behaviours, and assigning these to the most appropriate system. Computational intelligence learning techniques such as Artificial Neural Networks have been intentionally avoided, as these techniques commonly present their data in a "black box" system, whereas this work aims to make knowledge explicitly available to the user.

A biologically inspired memory system has further been proposed in order to generate additional behaviours in Artificial Agents, such as behaviour related to forgetfulness. This work explores how humans can quickly recall information while still being able to store millions of pieces of information, and how this can be achieved in an artificial system.

# DECLARATION

I declare that the work described within this thesis was undertaken by myself between the dates of registration for the degree of Master of Philosophy at De Montfort University, October 2008 to October 2012.

# DEDICATION

To Alex, for persevering.

To my parents, for believing.

# ACKNOWLEDGEMENTS

I wish to thank Dr Mario Gongora for his endless patience in supervising me throughout this MPhil. His ability to channel my thought process has been invaluable throughout, and without whom this thesis would not have been possible.

I also wish to thank the Institute of Creative Technologies, Professor Andrew Hugill, and the staff at the Institute who were helpful throughout, and funded this programme of research.

Finally my thanks go out to my friends and family for understanding and supporting me throughout the process.

# CONTENTS

# LIST OF FIGURES

# CHAPTER 1. INTRODUCTION

Artificial Intelligence in Serious Gaming is most commonly confined in use to generating "human-like" behaviour in artificial agents; a character that appears to be human is required to act as if they are human (with all the idiosyncrasies and flaws that this entails). The behaviour systems employed to perform this task have most commonly been derived from a single AI methodology, resulting in the constriction of the system to the disadvantages as well as the advantages of that methodology. It is a well-known problem in Games, serious or otherwise, that artificial characters are often prone to displays of just how un-human they are; running into walls, performing the same tasks repetitively, not learning from their mistakes or adapting to new situations in a way that a human would.

Existing behaviour systems offer convenient methodologies for producing behaviour; however they suffer from inherent limitations that this work aims to overcome through the use of hybrid methodologies. Finite State Machines (FSM) and Belief Desire Intention systems (BDI) are the primary candidates for hybridisation as they offer both quick-reactivity in the case of FSM, and dynamic and adaptable behaviour in the case of BDI.

While FSMs are quick to respond with a limited number of states, they can easily become overly large and complex to maintain, with an exponential growth in state transitions and processing time required when new states are added. Finite State Machines also suffer from being difficult to expand automatically, thereby lacking any practical method for automated learning.

BDI systems do not suffer from a growth in state transitions; however they can be slow to respond to a rapidly changing environment. By combining these two behaviour systems a hybrid system is proposed that is both fast to react and simple to maintain by separating behaviours into fast-reaction (instinctual and hard-coded) and slow-reaction (behavioural and adaptable) and applying these behaviours to the appropriate system. Computational intelligence learning techniques such as Artificial Neural Networks have specifically been avoided in this work, as they commonly suffer from issues surrounding the transparency of their data (black box systems), and additionally are complex to set up. This work aims to make it simple for a user of the system to begin the simulations with Agents with explicit knowledge in place.

Human behaviour is heavily dictated by our memories and the way in which our memories are stored and recalled; a previous experience in a situation informs further decision making when a similar situation is encountered, knowledge of the surrounding area influences path finding, etc. This has created an inspiration to construct an artificial memory system that attempts to emulate biological systems. A method is proposed whereby memories can be stored in an Agent and recalled in the future with varying degrees of success, thereby implementing behaviours such as forgetfulness without having to specifically program a behaviour to emulate this. In this work there is an examination of how humans can recall information quickly, while still being able to store millions of individual memory items and how this can be achieved this in an artificial system, retaining speed and capacity.

In searching for a way to overcome these limitations in existing AI methodologies, it can be shown that instead of adjusting or re-designing these existing methodologies to overcome their disadvantages, it would be more beneficial to have two or more concurrent AI methodologies controlling an artificial Agent, and swap between them while the Agent is executing, in order to utilise the most appropriate methodology for any given situation. Given that it would take an AI system by itself to determine which methodology was most appropriate at any one time, it is more effective to simply split the behaviour of an Agent into categories, and assign the most appropriate AI methodology to each.

At the most simplistic level, human behaviour can be split into two categories; behaviours that are instinctual, and behaviours that are learned over time. Given these two categories, the AI methodologies that best suit each can be determined. Instinctual behaviours are finite, but they require rapid decision making and limited deliberation time to determine which, if any, instinct should be active at any given time. In this work I propose that this task is most simply performed by a finite state machine, where each instinct is a state, with an additional, special, state for when the Agent does not need to respond in an instinctive manner. By contrast, learned behaviours are acquired over time, and require much more detailed processing. Belief Desire Intention systems are well suited for this task; an Agent's belief network can be updated over time, and his desires can be iterated over, paying attention to which desire is most critical at any given moment.

The complexity with this method comes from deciding how best to arrange behaviours. A full list of human instinct has never been assembled, and it may be impractical to do this, as many of our behaviours can be paradoxical as to their nature. A prime example of this comes from our behaviour relating to hunger and food. It would appear that humans have instinctual behaviour relating to hunger (as babies we cry when we are hungry), but also have learned behaviours as

well (as adults we learn to feed ourselves). In order to address this complexity, it has been necessary to allow learned behaviours to be triggered by instinctual stimuli, for example an Agent may enter an instinctual state of Hunger, but this triggers a learned behaviour relating to the abatement of Hunger. A further crucial part of behaviour is that of memory. A memory of a place or an object informs behaviour related to it; a memory of being stung by a wasp would cause behaviour to avoid wasps in the future. Memory results in our most complex behaviours, and it is therefore critical to be able to simulate memory in such a way that it is as close to biological memory as possible. Classical implementations of memory in artificial agents usually result in a system that has perfect storage and recall of information, and while this makes sense for such classical systems, when trying to simulate human behaviour, it is essential that we are also able to simulate the flaws in human abilities. One of the most obvious flaws in memory is that of "forgetting". Almost every human has the inherent ability to forget things that they once knew, or being able to only partially recall information that they once held with complete clarity, and these traits result in behaviours of their own.

Given the influence memory can have over behaviour, it is absolutely essential to be able to simulate memory in such a way that it is as closely analogous to biological memory as possible. By providing functionality that covers forgetfulness, flawed recall and context sensitive knowledge, it is possible to produce behaviour that closely mimics that of the natural world. Behaviours such as forgetting why we entered a room, where we left an item, or how to find something are only achievable through very specific programming, or by directly emulating the memory system and then building behaviours based upon the state of the memory model. Simulating "forgetting why I entered a room" in a believable manner is far more achievable by actually having the Agent forget, rather than having it pretend that it has forgotten. A hybrid behaviour system offers advantages over a single-methodology system by providing alternative methods when one methodology is not ideally suited to a specific task. By choosing the best methodology for any given task, the Agent is able to retain a believable behaviour as opposed to behaving in a way that would not be appropriate to its current situation.

This work has been primarily developed to assist in the creation of Serious Games, Virtual Worlds, and other Behaviour Simulations by providing Artificial Agents that can be easily created, modified and observed. These Agents can be examined while they are online and have their behaviour and knowledge adjusted in order to simplify the development of more realistic behaviours. The key aspects to this ability to customise the Agents are as follows:

- Agents can be initialised with pre-existing Behaviours and Knowledge
- Knowledge can be adjusted while the Agent is online

- Knowledge can be observed while the Agent is online

Artificial Neural Networks have been specifically avoided as another primary focus of this work is the construction of Artificial Agents with partially defined knowledge, and this is not easily achievable with ANNs. It is also a key aspect that the knowledge that each Agent acquires during its lifetime is subject to inspection by a user of the system, and ANNs are commonly a "black box" system, which prevents the information that they contain from being examined easily. These aspects are further discussed in Chapter 3 – Developing Behaviour Systems.

This work was funded by the Institute of Creative Technologies at De Montfort University, Leicester as part of their work on the Virtual Romans project (Hugill, 2012); an initiative to better understand the history of Leicester as a Roman town, Ratae Corieltauvorum. To this end the artificial agents constructed as part of this MPhil have been themed toward providing behaviour appropriate to Roman Britons, and artificial environments have been constructed with the same theme in mind.

This thesis covers the research undertaken to develop a hybrid AI methodology, using Finite State Machines and Belief Desire Intention systems, with a biologically inspired memory system forming the belief network of an AI Agent. This work offers the following contributions to knowledge:

- A scalable methodology for producing semi-realistic behaviour, capable of supporting a significant amount of concurrent behaviours.
- A Belief Network for BDI systems that is capable of adapting over time, and itself aids in the generation of behaviours by consisting of biologically-inspired features, such as the forgetting or distortion of Beliefs.
- A system that allows for the description of Agents in a natural-like language, thereby generating Agents that can be examined and understood by a human at run-time.
- A hybrid behaviour system that produces semi-realistic behaviour by selectively switching to the most appropriate behaviour system to deal with the current environment that the Agent is engaged in.

The following papers were published during the period of research described in this thesis:

- Storage, degradation and recall of agent memory in serious games and simulations (Irvine and Gongora, 2010a).
- Combined Artificial Intelligence Methodologies to Achieve a Semi-realistic Agent in Serious Games (Irvine and Gongora, 2010b).

- ReAd: reactive-adaptive methodology to enable evolving intelligence agents for virtual environments (Gongora and Irvine, 2010a).
- Adaptive intelligent agents based on efficient behaviour differentiation models (Gongora and Irvine, 2010b)
- Intelligent agents based on a hybrid adaptable structure and a biologically inspired memory (Irvine and Gongora, 2011a).
- Storing objects in a short-term biologically inspired memory system for artificial agents (Irvine and Gongora, 2011b).

# CHAPTER 2. LITERATURE REVIEW

## 2. INTRODUCTION

Artificially intelligent behaviour systems in serious gaming provide an area of research that deals with a wide cross section of AI theory. To achieve believable (i.e. not behaviour only achievable by a computer; perfect knowledge, flawless reactions, etc.) human behaviour in games requires the crafting of a behaviour system that is quick to react, yet produces behaviour appropriate to the situation. Behaviour systems now attempt to move toward Artificial General Intelligence (Strong AI) by solving multiple problems simultaneously, as opposed to Weak AI which favours specific problem solving. In this literature review I discuss the state of the art of AI techniques used to produce behaviour in artificial agents, examine the limitations in these techniques, and discuss where this work will further the research into novel methodologies.

## 2.1 BEHAVIOUR SIMULATION

### *2.1.1 FINITE STATE MACHINES (FSM)*

Finite state automata allow for the representation of state, and the transitions between such states to be inferred in an Artificial Agent. FSMs are used in current generation video games to simulate behaviour to a certain degree (Spronck et al., 2003) (Spronck et al., 2006) (Baumgarten et al., 2009) (Wood, 2004).

These state machines offer a limited amount of control of artificial agents, their limiting factor being that they cannot adapt or evolve; the behaviour is confined by what has been pre-programmed (or scripted). In addition, as the scripts necessary to simulate even low levels of behaviour are prohibitively complex, this can often lead to mistakes in programming, resulting in AI that is often flawed (Spronck et al., 2006) (Nareyek, 2001).

In most cases, NPC behaviour can be modelled in terms of mental states, therefore FSMs are a natural choice for game developers when designing NPCs (Yue and Byl, 2006). The programmer predefines each and every state. When the Agent enters this state it follows the rules laid down by the programmer. The Agent can then enter another state and follow another set of rules, or stay in the same state repeating rules infinitely.

Code complexity in a FSM increases significantly as the system scales up with more behaviours (Bartish and Thevathayan, 2002), and iterating through all the state options may become too

computationally expensive for a system required to run in real-time. If well thought out, FSMs can lead to convincing AI that can carry out tasks with some degree of success, however, as the amount of states increases, so does the complexity of the FSM. In this proposal, the aim is to use FSMs to deal with very specific states that can be considered necessary to be controlled directly to avoid behaviours that would be inappropriate.

### *2.1.2 Belief, Desire, Intention Systems (BDI)*

BDI theory was originally proposed in (Bratman, 1987) which describes a platform for a rational agent that allows for means-end reasoning, weighing of competing alternatives and interaction between these forms of reasoning.

Along with his original work, a further paper (Bratman et al., 1988) was later produced which provides key points of the theory, including the following:

- An adequate architecture for artificial agents must perform means-end reasoning and weigh alternative courses of action.
- This architecture must address the problem of "resource-boundedness".
- A rational agent is committed to doing what it plans.
- An agent's plans should be consistent.
- These plans should be partial.

In this state, BDI Agents provide a reasonable platform for the development of rational Artificial Intelligence, yet there are a number of issues that remain outstanding, some of which are addressed in their paper (Georgeff and Lansky, 1987). The paper describes a method for creating an action "stack" within a reasoning Agent. They present a situation where a robot is given a task, plans to carry out the task and begins the task itself, but it is then interrupted with new information and must evaluate what to do with the new information.

Ideally, Agents should be able to reason on the importance of information that they receive; some immediate danger must be addressed before a more mundane task is carried out, regardless of the order in which the information is received.

(Georgeff and Lansky, 1987) have suggested that the success of their Procedural Reasoning System "PRS" is due to the following factors:

- The PRS's partial planning strategy.
- The PRS's reactivity.
- The PRS's use of procedural knowledge.

- The PRS's reflective capabilities.

These factors indicate that a behaviour system should ideally construct its plans in part, and most importantly, at the last possible moment. This can be related to human behaviour with the following example:

> *"When planning a route from point A to point B, where the route is bisected by a road that must be crossed, does a person immediately plan where to cross the road (if there is no designated crossing point), or do they instead simply plan to get to the road, and then decide where to cross when they arrive there, and then plan the remainder of the journey?"*

It should also be noted that a rational Agent should be capable and willing to interrupt or abandon a plan when circumstances demand such action. For an Agent to retain or pursue a plan that a rational human would not clearly makes the Agent irrational.

Criticism of BDI Agents does exist; researchers commonly, based on their discipline, argue that the three attitudes of Belief, Desire and Intention are either insufficient or excessive. In my own opinion, I would note that parts of the human psyche such as instincts are not separated from any other using BDI; all aspects are combined under "Desire". (Rao and Georgeff, 1995) address this issue in their paper and argue that the three attitudes of BDI are necessary (although not adequate) in situations where real-time performance is required. (Rao and Georgeff, 1995) also mention that there has been "little work on bridging the gap between theory, systems and applications", and this is an observation with which I agree; while several papers exist that discuss the theory behind BDI, very few seem to provide an explanation of how such a system could be implemented.

To address this issue, Rao later worked on a formalisation of BDI as a language (Rao, 1996); an abstraction of an implemented BDI system such as the PRS earlier proposed by (Georgeff and Lansky, 1987). The author states that "implemented BDI systems have tended to use the three major attitudes [Belief, Desire, Intention] as data structures, rather than as modal operators", and aims to correct this by showing a "one-to-one correspondence between the model theory, proof theory, and the abstract interpreter".

Classic BDI systems "execute as they reason and are robust in handling failure, therefore well suited for complex and dynamic environments requiring real-time reasoning capabilities"(Silva et al., 2007). This indicates that classic BDI systems run in real-time and do not delay their reasoning in a way similar to that of human cognition.

(Silva et al., 2007) builds upon previous work (Sardina et al., 2006) that created the CANPLAN language, an extension of the BDI specification language CAN (Winikoff et al., 2002) by adding time limiting factors to the planning methods that CANPLAN describes. The method that the authors propose was noted to be relevant to this research project as it enables BDI agents to react in a similar way to that of human behaviour, allowing for a more realistic simulation. Every human has a varying planning time depending on the complexity of the task they are presented with, and using a pure BDI system that resolves problems immediately would not produce a realistic simulation. If for example an Agent was attempting to plan a route to catch a bus, it should not reason to the point where it is unable to reach the bus before it leaves. By restricting the length of time that an Agent spends reasoning about a problem, it is possible to generate more realistic behaviours.

BDI systems have evolved to a point where there are several programming languages (Rao, 1996) (Georgeff and Lansky, 1987) (Silva et al., 2007) that are capable of describing a BDI Agent. It has been suggested (Dennis et al., 2008) however that there are several problems with the popularity of the languages, namely:

- There are too many languages
- The languages are similar but subtly different, making it difficult to learn multiple languages
- Despite the fact that many BDI languages have logical semantics and utilise logical mechanisms, formal verification tools are rare

In an effort to resolve these issues, an intermediate language (Agent Infrastructure Layer or AIL) has been designed in order to:

- Provide a common semantic basis for a number of BDI languages, thus clarifying issues and aiding further programming language development.
- Support formal verification by developing a model-checker optimised for checking AIL programs.
- Provide, potentially, a high-level virtual machine for efficient and portable implementation.

It is noted that 3APL (Hindriks et al., 1999), AgentSpeak (Rao, 1996), and METATEM (Fisher, 1994) all use minor variations on first order literals for the representation of beliefs, goals, actions, etc. Jadex (Braubach et al., 2005) uses an internal JAVA representation but fragments of this can be mapped into first order logic. Because of this, AIL makes use of first order literals

as the basic representation. The authors of AIL note in their paper that key work in the field of verification tools needs to be undertaken, and as such more papers have subsequently been published to address the issue of developing a model-checker for BDI systems (Dennis et al., 2008).

More recent work (Patel and Hexmoor, 2009) has indicated that the current state-of-the-art in the field of Games AI results in bots (opponents) that are very predictable, with patterns that can be recognised across all current games. (Patel and Hexmoor, 2009) suggest a system using BDI Agents that produces more human-like, believable opponents that should be more interesting for players. They also note that BDI Agents are "computationally inefficient", but reason that with growing processing power this is less of an issue. While their reasoning may be sound I am more inclined to assert that the advances in modern processing power are a workaround solution to BDI's inefficiencies, not the long term solution that Hybrid systems offer.

### *2.1.3 Belief Systems and Agent Personality*

(Genesereth and Nilsson, 1987) differentiate Agent "beliefs" from knowledge by stating that knowledge "is simply that which is true in the world, while beliefs are an agent's understanding about the world". This indicates that what an Agent believes does not necessarily have to be the truth, that they can believe something that is in fact incorrect.

This difference is explored as a way to more closely simulate human behaviour in Agents (Bhargava and Branley, 1995), as humans often believe something that is not necessarily true. This paper notes that "In a computer simulation, an autonomous agent must receive its input through other computer programs or external sources; it has no sensors of its own to acquire information. If that is the case, and if it is desired that the agent does not receive perfect knowledge, how exactly are the agent's beliefs to be determined?"

In a traditional behaviour simulation, Agents are normally capable of perceiving the virtual world with 100% accuracy, and their behaviour is then based upon this accurate knowledge. Humans (or even robots operating in the real world) are not capable of always receiving 100% accurate information about the world in which they are operating, and we therefore have to make judgements about the information and determine if we think it is correct or not. Without this restriction Agents are incapable of making mistakes based upon incorrect information, and therefore they are not accurately simulating human behaviour.

To counteract this issue (Bhargava and Branley, 1995) have developed a method of simulating the belief system of an intelligent Agent, which allows the Agent to observe their virtual world in such a way that the information it receives is not guaranteed to be accurate. By implementing

this factor of error into an Agent's perceptions, the Agent is then capable of making mistakes, therefore instantly transforming its behaviour into appearing far more realistic.

In order for Agents to have semi-realistic behaviour, it is necessary for them to have a personality that is unique to them. Work in this field has been undertaken by (Ghasem-Aghaee and Ören, 2007) and describes how human personality can be recognised and identified using the "OCEAN" model of personality factors: (Howard, 2000), shown in Figure 1.

- O – Openness
- C – Conscientiousness
- E – Extraversion
- A – Agreeableness
- N – Negative emotionality

Figure 1. OCEAN personality factors

Each of these five factors has six facets, giving a total of 30 personality facets that can be modified for each individual agent. By generating Agents with different degrees of each facet, it is possible to create an Agent with a unique personality. It should also then be possible to compare these personalities against each other to determine which personalities are compatible and which are not, allowing Agents to form social groups, similar to the method used by humans.

A combination of Agent personality with simulated belief systems could result in Agents forming social groups with other Agents that are not necessarily compatible with their own personalities, resulting in a social group that will not function without conflict, adding to behaviour realism.

### *2.1.4 Fuzzy Logic*

Introduced in 1965 (Zadeh, 1965), Fuzzy Logic provides a method of defining a value between 0 and 1, or, the degree of "truth" for a value. While originally presented as a way of processing data into partial sets (as opposed to crisp set membership), it was later adapted for use in control systems, and therefore for the simulation of behaviour. By being able to define values in a more "human" manner ("slightly small", "very tall"), we can produce behaviour that appears more realistic.(Hellmann, 2001) (Zadeh, 1984).

The use of fuzzy logic in computer games is discussed in (Li et al., 2004) where they describe how varied and subtle behaviours from an artificial agent enhance "the perceived complexity,

enjoyableness and credibility of a virtual environment". (Li et al., 2004) go on to propose their hybrid system of BDI and Fuzzy Logic, by passing input from the virtual world through a fuzzy rule-based system they are able to generate more entertaining behaviours for their artificial agents.

(Bhargava and Branley, 1995) investigate the implications of constructing a belief system for an autonomous agent that closely mimics that of the real-world. In biological and robotic systems, input cannot always be guaranteed to be perfect; there is usually some degree of uncertainty, therefore to construct an artificial agent that behaves as if it is in the real world requires that it handles uncertainty. Consider for a moment the implications of uncertainty in artificial intelligence; if an artificial agent is faced with uncertainty at any point in its decision making process, it must resolve this uncertainty to some degree in order for it to continue. (Saffiotti, 1997) tests the claim that "the qualitative nature of fuzzy logic makes it a useful tool for dealing with problems characterized by the pervasive presence of uncertainty", stating that fuzzy logic techniques "have attractive features" when it comes to representing information "at the level of detail which is available".

### *2.1.5 Behaviour System Conclusion*

Many systems exist for the purpose of simulating behaviour in artificial agents, some of which are more suited to simulating human-like behaviour than others. In selecting existing systems to hybridise, it has been necessary to examine their applicability to simulating human behaviour, and as such, both BDI and FSM have been selected.

Finite state machines are commonly used in today's state of the art video games for the purposes of simulating behaviour, and as such have a proven track record in their ability to simulate human-like behaviour to a certain degree. However, several issues exist when implementing finite state machines, most notably a tendency to "flip" between states, or to become stuck in a particular state even when it is obvious to a human observer that another state would be more appropriate for the current situation.

Belief desire intention systems are built upon a foundation that was designed to bring rationality to decision making. Human beings have evolved their rationality on top of their underlying instincts, and this makes BDI systems an ideal candidate for simulating the rational behaviour found in humans. BDI systems do however have drawbacks; they are considerably complex to implement, they are not commonly capable of adapting their behaviour while online, and the simulations are more costly to run than that of a simpler behaviour system.

Fuzzy logic systems provide a strong methodology for describing uncertainty, and handling imperfection, however these systems need to be carefully constructed in order to achieve inputs that are similar to biological systems. A fuzzy logic based "sense" system would be a sensible addition; however it is outside the scope of this work which focusses primarily on the generation of behaviour through unreliable Beliefs which is handled through the biologically inspired memory system described in Chapter 4. Behaviour could be generated through the addition of "imperfect" senses, for example having an Agent be unsure as to whether it has seen something or not, and acting upon this by trying to become more certain. This is discussed again in Chapter 7 – Further Work.

## 2.2 HYBRID SYSTEMS

Hybrid systems cover the combination of two or more existing AI techniques such as Finite State Machines, Fuzzy Logic, or BDI into one combined system for the purpose of bringing together the benefits of all core techniques with fewer of the drawbacks.

(Rhalibi and Merabti, 2006) propose a hybrid system of Fuzzy Logic as well as Neural Networks in order to generate behaviours that can change over time depending on the input they receive. Their work describes in detail how fuzzy logic can be used to find the "degree of membership" of a variable within a set, allowing an agent to evaluate what action it should be taking at any given time. This method allows for the generation of behaviours that use linguistic variables to describe themselves.

While the authors of the paper were unable to make much progress in the form of testing, they did note that they did show how combining traditional AI techniques produced a system that "enables the choices made by a conventional AI layer to be altered in response to feedback from the actions selected", which clearly highlights the benefits of combining two AI systems together.

Fuzzy Logic has also been combined with BDI (Li et al., 2004) in order to enable the Agents to analyse their surroundings using Fuzzy Membership, enabling the Agents to group data into "sets" and then use these sets to make decisions with their BDI systems. It should be noted however that BDI requires a number of changes to deal with certain situations; "Commitment" to Intentions must be added to prevent an Agent from constantly changing its Intention, and Desires must be weighted to ensure more urgent Desires are met faster than less urgent Desires.

These required changes indicated that BDI by itself is an insufficient method of simulating behaviour, and must be modified in some way. By examining other technologies such as Finite

State Machines I noted that they are also insufficient as stand-alone methods; FSMs grow exponentially in their complexity with each State that is added, therefore they are unsuited to large Behaviour simulations. This drawback to FSM is discussed in (Bartish and Thevathayan, 2002), who implemented a simple design for a game, and then measured the code complexity of these implementations as both a Finite State Machine and a BDI Controlled Agent. Their research showed that as the amount of possible behaviours increased, "code complexity for the state machine increases significantly" whereas "for the agent [BDI] model, the complexity increases linearly". They also showed that the computation time for both behaviour systems increases linearly as behaviour counts increased, however "the overhead is much higher for the agents".

The results they present indicate that the FSM implementation is significantly more complex than the BDI implementation, and when a single additional behaviour is added to the design the FSM implementation increases in complexity by a factor of 8, with the BDI implementation only increasing by 1. In contrast, the paper also measures the speed at which the implementations can be run, and this shows that the cycle time for both FSM and BDI implementations increases in a linear fashion, however the BDI implementation has a much greater overhead, resulting in a slower implementation overall.

(Bartish and Thevathayan, 2002) suggested that research should be undertaken to determine if a combination of BDI and FSM would result in a less complex and faster running implementation, however it appears there have been few publications in this area. This is likely due to the techniques involved being considered outmoded, with researchers preferring to investigate newer computational intelligence methodologies.

## 2.3 REPRESENTATION OF KNOWLEDGE

In order for an Agent to store any sort of information, for later analysis or recall, it must implement some form of Memory System. Works that currently exist (Kuffner and Latombe, 1999) (Peters and O'Sullivan, 2002) concerning memory systems for fully formed Agents mostly deal with recording spatial memory in order to accommodate some form of navigation system, however there is research available (Brom et al., 2007) that addresses the problem of full episodic memory for Agents.

I address the concept of Knowledge Representation in the following section, (Irvine and Gongora, 2010)having chosen a biologically inspired memory system(Irvine and Gongora, 2010a) as my methodology for representing knowledge.

## 2.4 HUMAN MEMORY

In terms of simulating human behaviour in Artificial Agents, factors such as memory may not be an obvious point of concern; however it is my belief (Irvine and Gongora, 2010a) that human-like memory is critical in simulating human-like behaviour. Consider how often your behaviour is affected by your memory; if you forget your keys you go back for them, if you have a previous experience of an action you are trying to perform, your knowledge allows you to refine the process for the action. In addition, a memory system provides an Agent with their model of the world; where objects and places are located, paths for navigation, as well as a model of other Agents that it may encounter. To this end I believe it is necessary to construct an Agent that is capable of storing information about its experiences, however, before considering the problem of storing information in an Agent, it is necessary to examine how humans store their own information. As I am trying to simulate human behaviour as closely as possible, it is sensible to assume that attempting to closely replicate their biological systems will assist with this desired outcome.

### *2.4.1 MEMORY STORES*

In their paper, (Atkinson and Shiffrin, 1976) propose the multi-store model for human memory that splits its structure into three distinct parts, namely:

- Sensory Register
- Short-term store
- Long-term store

These different memory systems store information for different lengths of time, ranging from up to 500 milliseconds (Sperling, 1960) for the Sensory Register, to decades for the Long-term store (Atkinson and Shiffrin, 1976). The separation of short and long term memory had previously been discussed in early works on psychology (James 1890). Earlier work by Sperling, 1960 has provided indication that the Sensory Register puts a limit on the amount of information that can be recalled after the process of observing new information. Sperling, 1960 indicated that the Sensory Register was capable of holding approximately 12 items, but that these items degraded before all 12 could be reported back by the human subject.

The Short-term store features a similarly limited capacity to the Sensory Register, as indicated by Miller, 1956 who specified seven “chunks” (plus or minus two) as its average capacity. Page and Norris, 1998 and Baddeley et al., 1975 set the period of recall for Short-term store items without rehearsal (repetition) of anywhere from a few seconds up to one minute.

In the case of the Long-term store, it has been indicated (R. A. Bjork & E. L. Bjork 1992) that information can be stored indefinitely. Information enters the Long-term store after being rehearsed several times in the Short-term store. Atkinson and Shiffrin, 1976 indicate that the ability to recall items from the Long-term store may degrade over time, but the information itself remains relatively permanent. A visual representation of these stores is shown in Figure 2.

### *2.4.2 Memory Types*

The differing systems for storing memories are not the only concern when investigating biological memory, it is also necessary to consider the different types of memories themselves. It is generally accepted (Tulving, 1985) that there are three main types of memory, namely:

- Episodic
- Semantic
- Procedural

(Tulving, 1972) indicates that episodic memory "receives and stores information about temporally dated episodes or events, and temporal-spatial relations among these events." - indicating that episodic memory is an autobiographical form of memory. This type of memory is stored without rehearsal (i.e., the first time we encounter it, and we do not need to repeat it to reinforce the strength of the memory).

Semantic memory by comparison is memory of concept-based knowledge that is not related to a specific experience. Tulving describes it as the "memory necessary for the use of language". Retrieving information from semantic memory does not alter that information, but the act of retrieval creates a new entry in episodic memory.

Procedural memory is our memory of how to perform actions, such as riding a bike, or simply walking. This type of memory survives disuse extremely well, and the memories themselves do not need to be consciously recalled in order for them to be used.

### *2.4.3 Memory Degradation*

Once stored, memories in a biological system are subject to subsequent degradation before their eventual retrieval, (Estes, 1997) discusses this in his paper, presenting a possible model for memory loss. Research tells us that memory retrieval is "erratic, highly fallible and heavily cue dependant" (Bjork and Bjork, 1992), however "once entrenched in long-term memory, information remains in memory for an indefinitely long period of time".

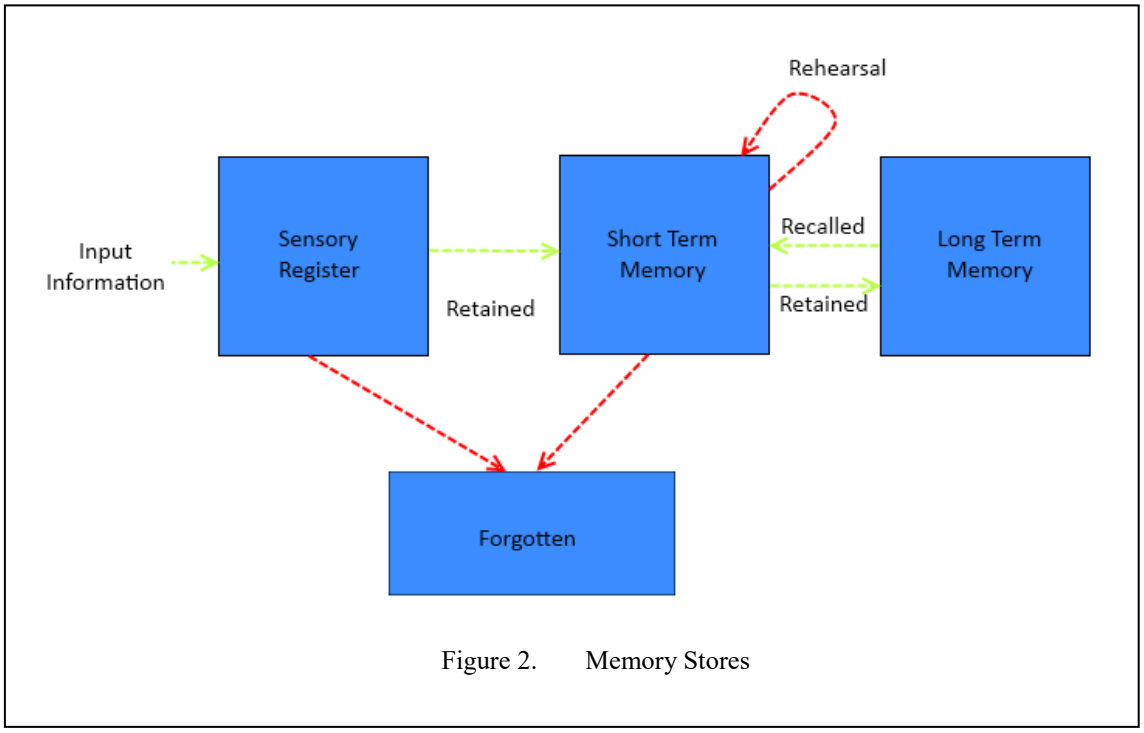


Figure 2. Memory Stores

Several papers (Brown, 1958), (Waugh and Norman, 1965), (Wickelgren, 1970) have been presented that contain psychological models for the process of forgetting information, however these have not been rich enough in structure to permit addressing many other aspects of memory loss (Estes, 1997).

Publications in the area of memory have classically remained relevant for an extended period of time, and only recently have there been more publications that can be more easily linked with the field of AI. It has been presented (Radvansky et al., 2011) that memory retention is linked to the environment in which we find ourselves, and movement within such a location, to such an extent that the phenomena of forgetting when walking through a doorway (memory declination due to location shift) has been extensively researched (Radvansky and Copeland, 2006)(Curiel and Radvansky, 2002)(Rinck and Bower, 1995). This may seem contradictory

It is also indicated (Radvansky et al., 2011) that memories can conflict with each other (competitive retrieval) to such a degree that even when the memories contain the same information, the very existence of multiple memories results in increased error rates in retrieval, and a slower retrieval time overall.

# CHAPTER 3. DEVELOPING BEHAVIOUR SYSTEMS

## 3. INTRODUCTION

As AI Agents in serious games progress forward, they continue to be primarily controlled by one AI methodology in any given implementation, be it Belief Desire Intention, Fuzzy Logic, or Neural Networks. While these methods function, applying them to an entire Agent often requires the method to be substantially modified, creating a very complex implementation. I believe that a hybrid system of several AI systems will result in a simpler, effective and efficient implementation and an Agent that behaves in a semi-realistic manner.

A BDI system often spends a high proportion of time in deliberation, and needs careful adjustments to ensure it has Commitment to an Intention, while preventing it from constantly changing to which Intention it acts upon.

Finite state machines can become far too large for a complex AI simulation; and are intrinsically designed to be fixed for the entire lifetime of the agent. An attempt to insert or delete states or state-transitions at run-time can be catastrophic or it can become an unfeasible or impractical validation problem from the algorithmic point of view.

In (Bartish and Thevathayan, 2002) the authors propose that research should be undertaken to determine if hybrid models of finite state machines and a BDI Agents' can reduce the design and coding necessary to create a behaviour system, while increasing the performance of such a system. There appears to have been no subsequent reports of such research undertaken in depth.

By combining these two technologies in a very specific way, I formulate a novel structure that in turn enables this novel methodology to create adaptable, complex and realistic Intelligent Agents more efficiently. In addition to making the system more efficient in terms of computing resources use, the structure can be enabled to be adapted at run-time so that the Agent can evolve and learn during its lifetime. With this approach, I will show that without compromising in any way the accuracy of the behaviour, it is possible to both significantly increase the performance and enable a behaviour based evolvable system.

In order to meet the requirements of an Artificial Agent that can have its knowledge adjusted while the Agent is online (as set out in the Introduction), the knowledge and behaviour stored in an Agent must be explicitly observable by the maintainer of the system; we must provide an

inner view of the Agent. This inner view should allow a user of the system to adjust the knowledge that an Agent has, such as providing the Agent with new Knowledge (and therefore Beliefs); for example giving the Agent knowledge of a nearby food source. This level of knowledge adjustment would allow a user of the system to easily comprehend the effects that a change to the Agent's knowledge would have.

It has been a continuous concern of this work to produce Agents that are "human readable", i.e. Agents that can be examined during run-time, and as such it has been necessary to rule out behaviour systems based on Artificial Neural Networks due to their "black box" effect. The knowledge that they contain is essentially locked inside and must be extracted. Research into this is currently being undertaken (Elizondo and Góngora, 2005) (Setiono et al., 2012) (Khan et al., 2012), however at the time of deciding upon a behaviour system, the "black box" effect of ANNs was unresolved and therefore offered an unacceptable risk in terms of developing a system that could be easily understood.

I take inspiration from how many creatures, including human beings, behave: at our core, we have innate instinctual behaviours and on top of that we have reasoning behaviours, which are learned from experience or example. Humans have then built upon these instincts through our evolution to create a method of reasoning for many other problems that require reasoning rather than merely reactions. From this it is possible to derive that a human's instincts are fixed throughout its lifetime, they themselves do not alter, although we are able to overcome our instincts with our learned behaviour. This separation between rational and instinctual behaviour has been used for some time in other fields such as robot control among others; Siciliano and Khatib, 2008 present many textbook aspects of robotics and in chapters 8 and 9 they present the reactive and the planning aspects which make use of instinctual and rational types of behaviour accordingly.

Our Agents are primarily controlled by a BDI system; this is my emulation of human reasoning. The virtual world in which the Agents exist is simulated in a crisp fashion; therefore it is necessary to simulate the belief system of the Agents, adding uncertainty and imperfection to their beliefs. These beliefs are passed on to the BDI system to enable the Agents to have a more realistic view of the virtual world. A FSM is given overriding control of the BDI system for situations where direct action must be taken with no deliberation, providing the emulation of human instincts.

## 3.1 EXAMINING BEHAVIOUR METHODOLOGIES

In order to identify the most appropriate methodology to produce realistic behaviour, it is necessary to examine existing behaviour methodologies. I first constructed a simple test scenario with which behaviour systems could be analysed for effectiveness and complexity.

### *3.1.1 EXAMPLE SCENARIO*

For this analysis, a number of Agents will be placed at random in a virtual environment; and each of these Agents has a basic need for social interaction. In addition to this basic need, they follow the instinctual desire to avoid outbreaks of fire. Each Agent is given 4 traits (Wealth, Education, Authority and Fitness), which will each be assigned a value between 0 and 1.

By varying the value of each trait that every Agent has, it is possible to create Agents that have different degrees of compatibility with each other. When created, each of the Agents is free to move about within the virtual environment at a certain speed. As the Agents move they look in front of them with a limited visual radius and focal length. Any other Agents that they encounter within this visual range they should approach and "interact" with. This interaction allows for a calculation of how compatible each Agent is with each other. This is a calculation based on comparing the value of each trait each Agent has, see Figure 3.

Compatibility = {

1 if $\delta Traits \leq 1$
0 if $\delta Traits > 1$

}

Where:

$\delta Wealth = Agent_1.Wealth - Agent_2.Wealth$

$\delta Education = Agent_1.Education - Agent_2.Education$

$\delta Authority = Agent_1.Authority - Agent_2.Authority$

$\delta Fitness = Agent_1.Fitness - Agent_2.Fitness$

$\delta Traits = \delta Wealth + \delta Education + \delta Authority + \delta Fitness$

Figure 3. Compatibility of Traits

If the other Agent is “compatible”, this Agent remains stationary as it has completed its task of finding a compatible Agent. If the other Agent is not compatible, this Agent moves off in search of other Agents. At randomly spaced intervals a fire breaks out at a random location in the virtual world. Any Agent near to the fire should then move away from it. Once they have moved away they return to their default searching behaviour.

This example was set in Roman Leicester (Hugill, 2012) as part of an early experiment into different methods of representation of a mapped environment.

### 3.1.2 Finite State Machine Based Model

Finite State Machines are an inherently simple behaviour system, and make an ideal starting point for analysis. This model demonstrates all the possible states that the Agent can enter in the test scenario, as well as determining the transitions that can occur to move the Agent between states.

As the Agents need to search the available space for other Agents to evaluate their compatibility, the Agent starts in the “Location Agents” state. While in this state the Agent enters a loop whereby it moves around the virtual environment waiting for other Agents to enter its field of view, and until this happens it will consistently transition back to this state via the “Can’t see Agents” transition. When an Agent does see another, it transitions into the

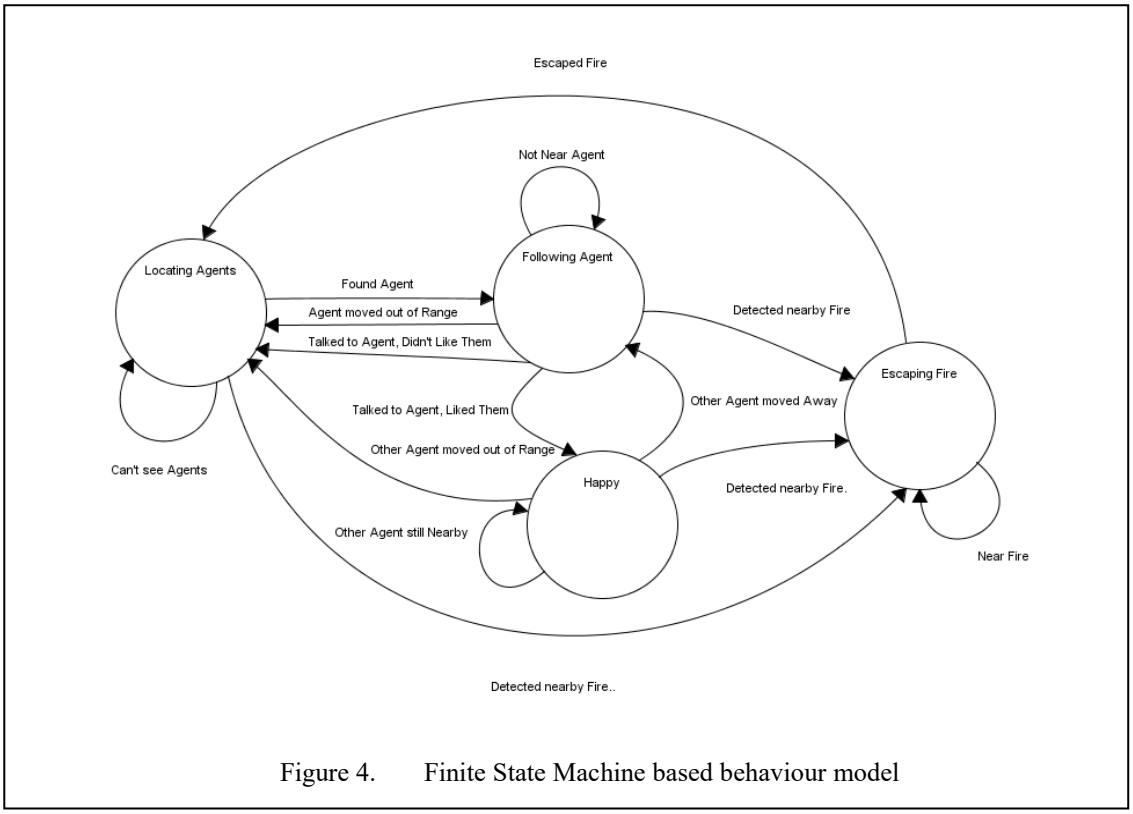


Figure 4. Finite State Machine based behaviour model

"Following Agent" state via the "Found Agent" transition. In order to compare compatibility with the other Agent it is necessary to be nearby, and as such, the Agent again enters a loop in the "Following Agent" state, transitioning back into this state via the "Not near Agent" transition. If the Agent loses track of the Agent it is following, it transitions back to the "Location Agents" state via the "Agent moved out of Range" transition.

When an Agent is finally able to compare its compatibility with another, it will then enter either the "Happy" state, or revert back to the "Location Agents" state. This occurs via the "Talked to Agent, Liked them" and "Talked to Agent, Didn't like them" transitions respectively. Once in the "Happy" state, the Agent again enters into a loop, governed by the "Other Agent still nearby" transition. If the other Agent chooses to move away in a direction that can be seen, this Agent transitions back to the "Following Agent" state via the "Other Agent moved Away", however if the other Agent moves away in a direction that cannot be seen, this Agent transitions back to "Locating Agents" via the "Other Agent moved out of Range" transition.

In every state it is possible to transition to the "Escaping Fire" state, and it should be noted that the addition of any further states would cause the model to become increasingly complex, as those states would also need to transition to "Escaping Fire". It should be noted that not all the possible states that the Agent should be able to enter have been included; without repeated tests of the scenario, not all possibilities may have presented themselves.

### 3.1.3 Belief Desire Intention Based Model

To implement the test scenario using BDI the Agent would need to have the following BDI components:

Beliefs

- Location of other Agents
- Compatibility with other Agents
- Location of Fire
    - Proximity to Fire

Desires

- Socialise with others
- Avoid Fire

Intentions

- Move to another Agent
- Talk to/determine compatibility with other Agents
- Move away from Fire

Figure 5. BDI Components

This model does not deal with the complexities of adding values of "Commitment" to each Intention. It should also be noted that the Intentions listed here are conceptual in basis, as a BDI Agent only has one Intention at any time, selected from its Desires. The priority of any Desire is also not handled; without this an Agent can follow a Desire blindly when another of its Desires should clearly take precedent.

While this model is not complex, implementing it highlights at least one of the potential limitations of such a system; namely that the computational cost of each BDI Agent is significantly higher than for a FSM implementation (Bartish and Thevathayan, 2002).

### *3.1.4 Model Analysis*

#### 3.1.4.1 Finite State Machine

Implementation of the Finite State Machine behaviour model revealed shortcomings with the methodology that I had not previously considered, namely that of inefficiency of implementation. As I progressed with implementing the FSM, time was increasingly spent on tweaking the States, and the transitions between them in order to cause the expected behaviour. Figure 6 shows a number of Agents following others in a form of "train", a behaviour that was both unexpected and unwanted. It was necessary to spend a long period of time altering the States of the Agents to ensure that this form of behaviour did not manifest.

The Finite State Machine implementation had a fast execution time, with each loop of the application taking less than a millisecond to execute. The testing was conducted on a modern machine, so it may well have been necessary to introduce many more States and Transitions in order to tax the computer in such a way that a more accurate reading of run-time could be ascertained. With the number of States concerned, code complexity was still relatively high. Each State was permitted to handle its own Transitions to other States, however a refactoring of the code so that Transitions were computed at the end of the application loop in one chunk, rather than performing similar checks in every State, would have reduced the amount of code significantly. In addition to this, it was necessary to break out of the processing loop after a transition was made in order to ensure that an Agent did not get to process two States in a single loop depending on their organisational structure in the code.

With the FSM implementation there did appear to be an unavoidable issue with code repetition. Different States required access to different information (nearby Agents, nearby Fires, etc.); however some States required access to the same information. In order to keep the FSM efficient, it made more sense to access this information in the States that needed it, rather than

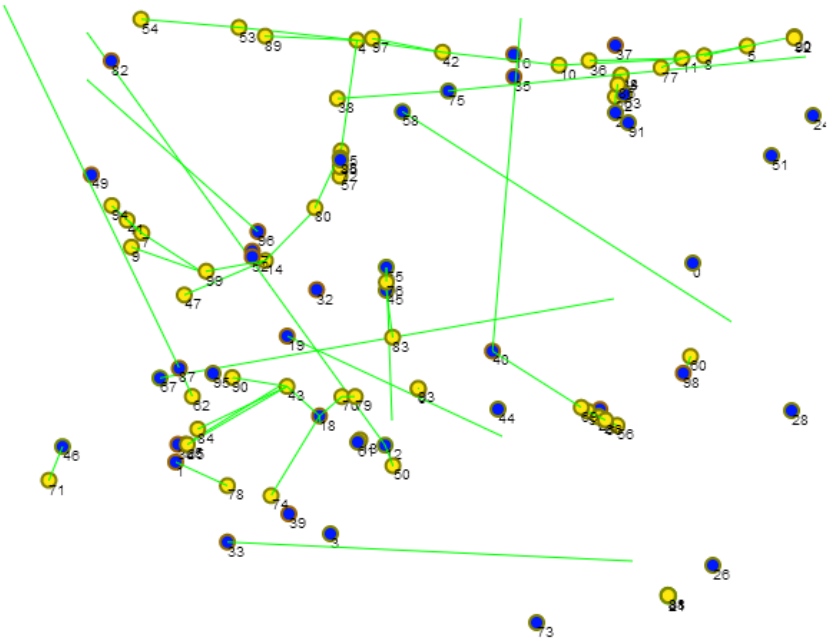

Figure 6. Finite State Machine based Agents during run time

accessing it generally at the beginning of each application loop. However this meant that several States accessed the same information in the same way, driving up code complexity.

The FSM implementation also lacked any method to provide commitment to a State. As an Agent transitioned State, all information about the previous State was lost and was not recoverable if the Agent transitioned back into the first State. This lead to one of the key shortcomings of FSMs for behaviour simulation; unrealism. As the Agents move around the virtual environment, they would randomly choose a target location, but if their "travelling" State was interrupted, the Agent would lose all knowledge of their target location, and upon returning to the "travelling" State would have to choose a new target, often resulting in an Agent backtracking.

### 3.1.4.2 Belief Desire Intention

The implementation of the Belief Desire Intention behaviour model proved to be significantly more complicated than expected. While the actual Desires and Plans are relatively simple, there is a lot of "framework" that had to be implemented first. It is therefore difficult to determine if the BDI system is actually less complex than a FSM. Adding further behaviours to the BDI system may reveal if it is ultimately simpler to maintain than a FSM.

The BDI system had a similar execution time to the Finite State Machine. Again it may be necessary to increase the amount of Desires to gain a fair assessment of execution time against a

FSM. The addition of new Desires should be less complex than in the FSM, as Transitions do not need to be considered. Transitioning between Desires is handled entirely by performing a state comparison for each of the available Desires and picking the Desire that is most applicable to the current Agent and World state, and so instead of calculating if a Transition needs to occur after a Desire has been chosen, each individual desire is informing the BDI system how applicable it is.

In order for additional Desires to be added to the BDI system, it was necessary to further abstract some key concepts in the simulation. Desires needed to be given "Expectations", and Plans needed to have "Consequences". Each Consequence is able to calculate if it fulfils a given Expectation, allowing for the comparison of applicability of each Plan to a specific Desire. Consequences and Expectations both have "Effects" which consist of an operation on an "Attribute", which can be either an Attribute of the Agent, or of the Virtual World. These additional abstract concepts initially add extra complexity to the code necessary to perform the simulation, but provide a more maintainable framework for adding further Desires and Plans.

## 3.2 Separation of Behaviour

There has been recent research in improving the performance of AI implementations for games in terms of more efficient use of resources (computational performance); in particular the work by (Osborne and Dickinson, 2010) uses a hierarchical approach to present dynamic detail in behaviour of the agents, depending on how near they are to the player.

When we take an instinctual and rational behaviour system and convert it to the closest matching AI technologies, we are presented with the following methodology: The Agents are primarily controlled by a BDI reasoning system. As the virtual world in which the Agents exist is simulated in a crisp fashion, it is necessary to simulate the belief system by explicitly adding uncertainty and imperfection to their beliefs; this will be addressed in section 3.2.1. These beliefs are then passed onto the BDI system to enable the Agents to have a more realistic view of the virtual world. At the core of the behaviour systems, a FSM is given overriding control of the BDI system for situations where direct action must be taken with no deliberation, providing the emulation of instincts.

### *3.2.1 Belief simulation*

It is important to remember that an autonomous agent's decisions and plans are influenced by the information it receives about the world in which it is operating (Bhargava and Branley, 1995). For the purposes of attempting to accurately simulate real-world behaviour, a

purely computational agent is at an immediate advantage; without the additional stage of filtering data that the agent receives from the Virtual Environment, the agent is automatically given a cognition of a situation that no real-world creature is capable of obtaining—our Agents automatically ‘‘see and understand everything’’, and as such it is necessary to inhibit this ability.

Work on simulating belief systems of agents has already been undertaken by Bhargava and Branley, 1995. They suggest that when simulations are used for real-world decision support tools, it is important that the agent’s belief processes are also properly simulated, i.e. based on realistic observation and detection techniques.

The authors in Bhargava and Branley, 1995 also suggest ‘‘conceptualizing the agent’s activities in terms of three main steps: detection, measurement and interpretation’’. This idea has been carried over to the belief simulation in this work. As the Agent detects new information, this should be measured in such a way that the level of accuracy varies depending on the state of a particular situation. If for example, the Agent is supposed to ‘‘see’’ other peers, the capacity to recognise or identify specific characteristics in them must be inversely proportional to the distance at which the observed peer is from the Agent. Once a level of accuracy has been determined, it is possible to affect the quality of information that the Agent receives. This can be defined as the ‘‘interpretation’’ stage.

Given a set of imperfect data the agent must make a decision on how to interpret it. If an object is visible in the distance to the Agent, it is less likely to interpret it correctly as opposed to a nearby object. By knowing how imperfect data it has received is the Agent is capable of making a decision of what to do with the perception it can create. This aspect of the decision making process is handled by the BDI component of the methodology.

The obvious choice to implement this perception system is to use Fuzzy Systems; we can enable evolvable membership functions to adapt towards a realistic simulation in various ways, as reported by the research community (Angelov, 2002) (Caises et al., 2010).

A particular alternative for the implementation of an adaptable and evolvable knowledge interpretation (and belief representation) component is to use a constructive dynamic memory structure devised for this purpose (Irvine and Gongora, 2010a). This structure allows knowledge, as observed by the agent from its environment, to be added dynamically and accessed by a series of dynamic strength links, with the link strengths performing the task of rendering knowledge “imperfectly”.

### *3.2.2 PLANNING AND DECISION MAKING WITH BDI*

Decision making via a BDI system enables us to control agents' higher level functions, such as social interaction, complex problem solving, and other behaviours that are not otherwise covered by human instinct. The authors in (Rao and Georgeff, 1995) proposed an abstract interpreter for BDI:

```
BDI-interpreter
Initialize-state();
repeat
        options := option-generator(event-queue);
        selected-options := deliberate(options);
        update-intentions(selected-options);
        execute();
        get-new-external-events();
        drop-successful-attitudes();
        drop-impossible-attitudes();
end repeat
```

Figure 7. BDI Interpreter

*"In this basic interpreter, the agent updates its beliefs, and then updates its available options. Once it has done this it examines its desires, matches them against these options and proceeds to deliberate over them, before updating its intentions and executing the plans to facilitate those intentions. In the final steps, the interpreter removes successful and impossible goals and intentions."*

By expanding on this abstract interpreter I am able to form a method that allows the Agents to make decisions about the virtual world based on imperfect values it receives, and then form plans to determine its actions in this virtual world. Methods to expand the abstract interpreter already exist in several implementations, including JADEX (Braubach et al., 2005), METATEM (Fisher, 1994) and AIL (Dennis et al., 2008). These existing implementations provide a software framework for the creation of goal oriented agents (Braubach et al., 2005), and are suitable to implement BDI in the hybrid system. Because of the algorithm required for implementing BDI, Agents can suffer in the aspect of realism due to their rational- focused nature. If not carefully controlled, a rational Agent can, in some cases, continue a behaviour that could not be considered rational or predictable in real terms.

In a situation where an Agent is trying to achieve a Desire with a high priority, it may ignore a situation that would naturally have a higher priority. By being too rational-focused it can appear to be irrational in cases where critical behaviours are ignored by more ‘‘desirable’’ ones. While this issue can be overcome with a complex over prioritised implementation, it can be avoided entirely by extracting behaviours that could be considered ‘‘priority behaviours’’, or ‘‘instinctual behaviours’’ from the BDI rational scope. By handling these behaviours outside of BDI, it is possible to avoid the potential problem of the Agent ignoring them.

### *3.2.3 Finite State Machines for Instinct Emulation*

Instinct is often an overriding factor in behaviour. In situations that are life threatening, instinct comes into play and determines actions during the situation with little reasoning or thought for desires or consequences. While instincts can be controlled, it is more common (and expected) to let an instinct take hold and dictate the actions of the individual.

As explained in the previous section, while a BDI system is potentially capable of simulating instinct using a prioritisation pre-programmed plan structure, due to the nature of BDI it is often costly to run this simulation with respect to the rational situations. A FSM is more efficient at handling and deciding on predefined behaviours, equivalent to instinct. Finite State Machines are capable of making an Agent act a certain way in a finite amount of states with little processing requirement, therefore making them very fast when the list of states is small.

The larger a FSM becomes, the less manageable it is for the system’s programmer, and the more states and state transition conditions the Agent must iterate through to determine its decision. For this reason, in this methodology, a FSM should only be responsible for the states that have been defined as being ‘‘instinctual’’ as shown in Fig. 7. The structure contains a small set of basic instincts, dealing with very specific cases; basically a single rational state will be defined, and a small set of instinctual states around it. After the Agent receives data from the knowledge interpretation stage this is passed on in the first instance to the BDI belief’s system for interpretation. Once evaluated, the perception data is routed to the FSM, which is then able to judge the state that the Agent should be at the current time. If the Agents’ state is recognised as ‘‘instinctual’’, the FSM keeps the control from the BDI and completes the whole decision cycle in the appropriate instinctual state.

Once the Agent’s state has changed to rational, the FSM passes control to the BDI, where the belief simulation system generates a new set of beliefs and continues the reasoning process to decide on the final plan and subsequent actions.

## 3.3 Constructing A Hybrid Method

When analysing this hybrid BDI-FSM system it can be seen that combining AI methods produces a technique wherein methods that are better suited to simulating behaviour in certain situations. As I discussed in section 3.1.4.2, additional abstractions have been added to the BDI system in order for the simulation to be constructed, however this leads to an additional overhead in terms of processing time. When the amount of possible Desires reaches are sufficient number, it is conceivable that the amount of time required to process all current desires would exceed the amount of time needed to react in a timely manner to a situation that the Agent may find itself in.

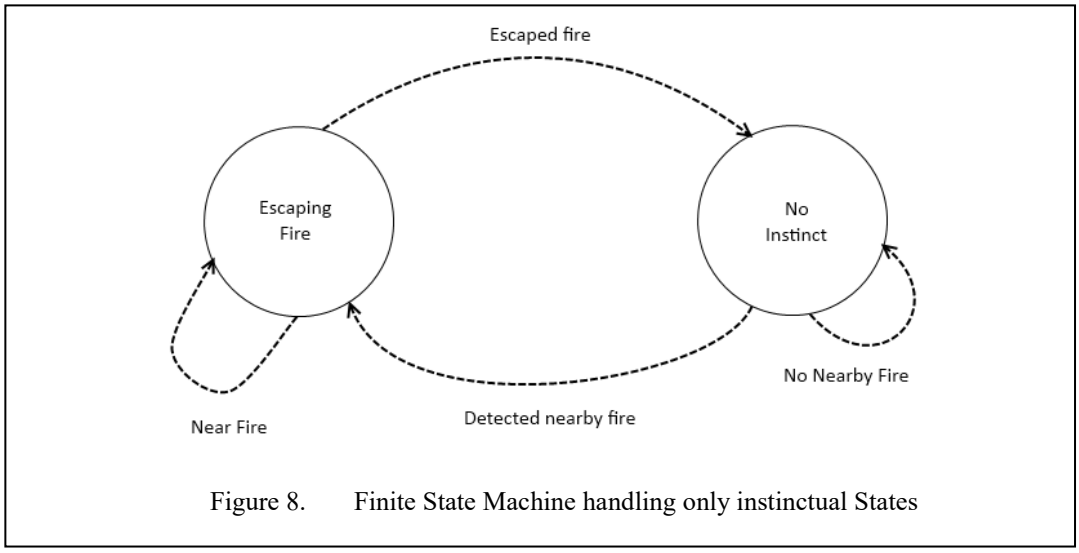


Figure 8. Finite State Machine handling only instinctual States

Even if Desires were given a priority, the entire list of possible Desires has to be iterated in order to find all the Desires that are applicable in the current scenario. If this list is of sufficient size, even high priority Desires may not be found quickly enough to act upon them. To combat this I propose a separation of the highest priority "Desires" (Instincts) into a separate Finite State Machine.

By separating the behaviours with the highest priority, these can be handled entirely without the processing overhead of the BDI system, thereby removing the need to include them in the list of Desires that is iterated to determine which Desire should be executed. Each Desire is converted into a State, with appropriate Transitions to other States, or with a Transition to the BDI system. These States can then be transitioned to without needing to evaluate which of these Behaviours has the highest priority.

### *3.3.1 Proposed Hybrid Model*

The hybrid model will combine the most appropriate components from both FSM and BDI in order to reduce the overall complexity of the system as well as reducing the computational cost of running the test, while providing a framework for learning and evolving behaviour.

This FSM component deals with situations where it is believed an "instinctual behaviour" is necessary to resolve the situation. In this case the need for an Agent to avoid nearby fires has been defined in this category.

In this model, the BDI system handles any other situations when the FSM is in the "Not Escaping" state.

*Beliefs*

- *Location of other Agents*
- *Compatibility with other Agents*

*Desires*

- *Socialise with other Agents*

*Intentions*

- *Move to other Agents*
- *Talk to/determine compatibility with other Agents.*

Figure 9. Proposed BDI Model

### *3.3.2 Analysis of Hybrid Model Effectiveness*

I have observed that using FSM as a method to produce Agent behaviours can result in a system that quickly becomes complex to implement and manage. Every state must be pre considered, as well as every possible transition between states. As noted in section 2.1.1 any additional states increase the complexity of the model significantly, requiring multiple transitions to other states. However, runtime performance for a FSM system compares favourably to that of a BDI system, especially when the amount of concurrent Agents being simulated is increased to a significant amount. This analysis of hybrid models is explored further along with the potential for creating evolving Agents in an earlier paper (Gongora and Irvine, 2010a).

Our BDI model maintains a low complexity when compared to the FSM design, and adding additional behaviours does not affect this. However, as noted in section 3.2.2, I have not included any information regarding the Agents commitment values or its priority values for plans; without this planning, Agents using the BDI model are subject to situations where they react in a unrealistic manner. The BDI system also suffers from an increased computational cost as the amount of Agents increases.

In this combined model, the FSM method becomes significantly less complex, requiring only two states to handle the one "instinctive behaviour". When the FSM is in its "Not Escaping" state it passes control to the BDI system, which is also less complex as it has fewer Desires. By keeping control of instinctive behaviours in the FSM, it is possible to avoid the higher computation cost of a BDI Agent and react more quickly to situations where it is deemed necessary, without the requirement of deliberating over a BDI system's Desires and checking commitment and priority values.

### *3.3.3 REPRESENTING BELIEF NETWORKS*

In the following sections I attempt to address the limitations in human-like memory simulations, providing a method for both memory degradation and long-term recall that enables an Intelligent Agent to more closely mimic a biological system, as well as having a more realistic view or model of the world.

By developing a biologically inspired memory system to act as the Beliefs in the BDI methodology, I provide a Belief network that acts in a more realistic manner, providing important behavioural cues such as forgetting, imperfect recall and location-sensitive memory. With a Belief network that is actively updated while the Agent is online, the Agent is capable of developing or adjusting its existing Beliefs to more accurately represent the virtual world. This novel contribution further enables the creation of evolvable and adaptable BDI systems, as their Belief network is no longer fixed; it adjusts and expands as the Agent further explores the virtual world.

# CHAPTER 4. BIOLOGICALLY INSPIRED AGENT MEMORY

## 4. INTRODUCTION

In order to create artificial Agents that behave in a realistic manner, having them emulate aspects of biological life in as many ways as possible is useful. This can be achieved by simulating the belief system of the agent so that it has a more realistic (i.e. less perfect) understanding of its surrounding environment (Bhargava and Branley, 1995), or providing the agent with a realistic method to store information that it acquires during its lifetime. In this chapter I present a novel biologically inspired memory system, with particular attention to the short-term storage methodology described herein. The initial structure of this methodology was proposed in (Irvine and Gongora, 2010a).

For an Agent to have a working belief system, it is necessary to be able to store the Agent's knowledge in such a way that information can be retrieved quickly. As I will discuss, it is also necessary to provide a method of obfuscating an Agent's knowledge so that it is not possible to retrieve memories with total accuracy all of the time; an Agent with an imperfect memory system will be able to simulate more realistic human behaviours by displaying traits such as forgetfulness. For example, an Agent that can clearly recall all of its knowledge is not realistic (although effective); normal humans cannot achieve this. To provide this imperfect memory system it is necessary to describe a memory model that allows for parts of the Agent's knowledge to be inaccessible, while other parts remain. This type of behaviour is not inherent in computer systems, and as such it is necessary to build a framework in order for this to be performed in as natural a manner as possible.

It has recently been discussed that artificial Agents should have the “capability to remember, reconstruct and forget information perceived from their interaction environment”(Lim, 2012) as this will “help them to comprehend their world and act reliably, learn context- and socially appropriate behaviour, focus their attention on important information relevant to the current interaction situation, make predictions about potential actions, learn from experiences and avoid repetitive behaviour”. These aspects are all key to the goal of realistically simulating human behaviour.

The ability to recall every single piece of information ever encountered with absolute clarity is not something achievable by most humans, with the notable exception of those with eidetic

memory. It is to be noted however that information learned by a human is believed not to be lost, but rather becomes non-recallable with disuse (Estes, 1997).

During this research I have evaluated some of the biological systems that are in place in humans, and attempted to theorise on a method to simulate these systems in artificial agents (Irvine and Gongora, 2010a). In this chapter I will discuss how these systems can be used in an artificial agent to store knowledge it obtains in a virtual world.

## 4.1 SENSORY AND SHORT-TERM MEMORY

Sensory memory and Short-Term memory are distinctly different memory stores. Sensory memory commonly has a much larger capacity than short-term memory, but a far shorter storage time (Sperling, 1960) (Phillips, 1974).

### *4.1.1 SENSORY MEMORY*

When information arrives for processing in biological systems, it is temporarily retained in the Sensory Memory store. In the case of visual stimuli, this is referred to as Iconic Memory (Sperling, 1960). This memory store is able to contain large amounts of information for a very short period of time, usually less than a second, and no more than two.

### *4.1.2 VISUAL SHORT-TERM MEMORY*

As this work is dealing with the storage of objects that an artificial agent has encountered and have been recognised by the agent's vision system, and as studies have shown that short-term memory is separated for visual and verbal memories (Baddeley, 1992), the Agents' Short-Term Memory is in the form of VSTM (Visual Short-Term Memory). VSTM is distinct from Sensory (Iconic) memory in that Sensory memory has a much larger capacity, but much shorter storage time; approximately 250ms (Sperling, 1960)(Atkinson and Shiffrin, 1976).

Research suggests that the capacity of VSTM is very limited, up to a maximum of three or four items (Atkinson and Shiffrin, 1976)(Miller, 1956)(Bjork and Bjork, 1992), however there remains a great deal of argument as to what constitutes an "item" (Lee and Chun, 2001). This issue is discussed further in section 4.5.

## 4.2 LONG TERM MEMORY

Long-term memory is the largest and most robust memory store. Information can be stored here indefinitely (Bjork and Bjork, 1992). Once information has been rehearsed several times in STM it can move into LTM and become more permanent. The ability to access long-term memories can degrade over time; however the information itself is relatively permanent.

### *4.2.1 Memory Loss and Retrieval*

From personal experience we all know the frustration that can come from "tip-of-the-tongue" moments, and other times when we fail to recall information that we once knew well (Atkinson and Shiffrin, 1976). Research tells us that memory retrieval is "erratic, highly fallible and heavily cue dependant" (Bjork and Bjork, 1992), however "once entrenched in long-term memory, information remains in memory for an indefinitely long period of time".

If memory is stored indefinitely, then why is it that we seem to have such trouble remembering? It would appear that items of information we have once learned, with disuse can become non-recallable (Bjork and Bjork, 1992), and that this facility is in fact beneficial and essential; because our storage capacity is so large we must loose retrieval access to that which we do not use.

A number of psychological models for the process of forgetting information (degrading memory traces) have been suggested (Brown, 1958)(Waugh and Norman, 1965)(Wickelgren, 1970), however these models have been implemented in average curves of forgetting (fading). In no case have they been rich enough in structure to permit addressing many other aspects of memory loss (Estes, 1997), for example they do not allow for the occurrence of spontaneous recovery of memories.

A method has been put forward whereby every item in memory is given two "strengths", namely a storage strength and retrieval strength (Bjork and Bjork, 1992). I aim to mimic this system closely with my form of memory retrieval. These strengths, respectively, indicate how well learned an item is and the current ease of access to the item in memory. It is assumed that storage strength can only increase, whereas retrieval strength can vary.

### *4.2.2 Loss and Forgetfulness in Existing Systems*

In some systems, it is likely that having an Agent forget information would be useless to the system itself, as the Agent's main purpose could be to gather information (spatial mapping applications, knowledge based systems), for this reason I have focussed mainly on Artificial Life simulations as those most commonly associated with Agents that need to forget learnt knowledge.

Full memory retention remains uncommon in simulations, however research has been conducted into the field of Full Episodic Memory (Brom et al., 2007)(Nuxoll, 2007). This research has indicated that forgetting is "essential" (Brom et al., 2007), due to the sheer amount of data recorded by an Agent. While it can be accepted that an Agent requires a working

maximum to its memory size, due to current day RAM requirements during simulation, a system could be designed so that LTM is cached to and from large capacity storage (database, disk, network share). We should also accept that not all data captured in the SM store should be kept, as this would require virtually unlimited capacity, and even in biological systems this method of data rejection is used (Atkinson and Shiffrin, 1976).

Other systems concerned with memory in Artificial Life also suggest that some knowledge should be forgotten in order to keep knowledge current, commonly the case in foraging simulations (Hirvonen et al., 1999)(Dumont and Hill, 2001). While this may be appropriate for some biological life, we know that in humans, "forgotten" memories are subject to later recovery, albeit with the possibility of distortion (Estes, 1997).

The field of Serious Games can differ somewhat in its use of AI from that of standard computer game systems. While a computer game opponent may know the location of its target at any time without the realistic use of input, this sort of behaviour is unacceptable in a Serious Game trying to simulate reality.

As mentioned in (Brom et al., 2007), in order to achieve realism in certain types of games, NPCs must be simulated even when the player is not in the location, and they must be equipped with full episodic memory. When we combine this with the fact that to fully achieve realism an Agent should never loose data, it can be concluded that an Agent must store all pertinent information in a way that can be recalled, subject to "failures of retrieval" (Anderson and Schooler, 1991) and distortion (Estes, 1997).

With this in mind I find that there appears to be no system capable of combining these requirements. When modelled on biological memory systems other than that of a human, it appears all AI methods must forget data permanently to some extent; even ant-pheromone trails fade over time (Parunak, 1997).

Remembering something that has long been forgotten is a fascinating aspect to human memory, and remains something that has not been simulated in Serious Games. Consider the added realism of interacting with an Intelligent Agent who without pre-scripting has forgotten something, but is then able to recall it with the proper cue from the player.

## 4.3 Storing Objects in Intelligent Agents

In this section I discuss the basic processes involved in encoding an encountered object for storage in an agent's memory system. I provide an overview of the process, describe how objects can be stored in iconic memory and how the features of an object can be extracted and encoded for storage.

Given what we know about short-term memory as a biological system, this can be applied to an artificial agent in order to allow the agent to encode and store any object that it encounters in a virtual world. Unless otherwise specified, further references to object storage and short-term memory refer exclusively to the proposed method in an artificial environment. When considering the implementation of this system, I define a "cognitive processing frame" as the time provided to an artificial agent to perform its cognitive processing; commonly one loop of an agent's execution cycle. This time is commonly assigned by the simulation that contains the artificial agent. While there is no direct biological equivalent, a processing frame could be most closely linked to every time a decision is made in a biological system. These decisions can range from a complete behaviour change, to a low-level behaviour such as involuntary motor movements, "considering" existing knowledge, or evaluating a surrounding environment.

### *4.3.1 Process Overview*

Storing an object in short-term memory is a multi-stage process that handles the examination and encoding of the object, as well as the actual storage. We assume that object capture is sufficiently handled by some other system that passes object definitions to the artificial agent as described in definition (1) below.

### *4.3.2 Objects in Iconic Memory*

When an object is encountered by the agent, a definition of this object is passed to the agent's cognitive system. Research tells us that this form of sensory memory is high capacity, with an implied upper limit that is, as yet, unknown (Phillips, 1974). Iconic memory is also constrained by spatial position of the objects it contains, and has a storage time of about 250 msec. To simulate this, I propose the use of a standard programming container (vector, list, etc.) for holding an unlimited number of objects. The entire object list is then processed in one frame of cognitive processing.

Object {ID, Position, Attributes [Attr1, Attr2…]} (1)

As objects are passed to the cognitive system of the agent, they are to be stored in a standard container that is accessible in the next processing frame. This allows an agent to determine

which objects it processed in the preceding frame, eliminating repetitive analysis, encoding and storage of objects.

### *4.3.3 Attribute/Feature Extraction*

As each object is processed by the agent's cognitive system, each attribute or feature of the object is extracted from the object itself. Agents have no way of distinguishing objects between each other except for their attributes, however in the case of identical objects, the only differentiating attributes are the object's positions; only one object can be in one physical location at a time.

As there is only one way to determine the difference between objects it is important that the position of the object is the first to be extracted from an object as it is the unique identifier for every visible object in the scene. After the extraction of an object's location, all other features can be extracted in any order.

It is important to note that Attribute and Feature extraction can be performed on any part of the virtual world that has been properly described, not merely Objects that the Agent encounters. This means that using this method, an Agent would be capable of analysing the appearance of other Agents and storing their Attributes/Features in memory, as well as analysing and storing events that occur that the Agent witnesses. This ability would then further enable the Agent to:

- Store and recall the identity of individual Agents (assuming Agents are created with differing Attributes)
- Store and recall the context in which Knowledge was acquired (for example, storing the location at which a food source was discovered)
- Store and recall locations where interactions commonly recur (for example, a given Agent can often be found in a given location)

### *4.3.4 Feature Encoding in Visual Short-Term Memory*

With a list of features extracted from an object, the cognitive system must now store these features in some way, namely moving them into VSTM.

Each feature of an object is then encoded into a memory chunk; a generic container for both entire objects, and individual features as described below in (2). These chunks are then moved into VSTM for short-term storage. In previous work on biologically inspired memory (Irvine and Gongora, 2010a) I have used the concept of chunks to be able to store an undetermined number of any data-type permanently in Long Term Memory.

As these chunks are not referenced explicitly, it is necessary to create an overall "container" for feature chunks, and a chunk for the object itself is ideal for this task. An object chunk is created, and then has all the feature chunks linked to it, with each feature chunk also linking back to the object chunk. Each feature chunk is now added to VSTM. Section 4.5.3 discusses my reasoning for storing features individually.

Chunk {Type, Creation Time, Data, Links [Cnk1, Cnk2…]} (2)

## 4.4 USING MEMORY SYSTEMS FOR WORLD REPRESENTATION

Memory use in the system is critical to the way the agent sees the world and therefore how realistic the simulation is. It has previously been discussed that "Creatures with subtle behaviours enhance the perceived complexity, enjoyableness and credibility of a virtual environment" (Li et al., 2004) and that "fuzzy rule-based systems allow the nuances among inputs to be captured and further reflected in the decisions" (Li et al., 2004). It is clear then that a nuanced representation of the virtual world in which the Agent resides should allow the Agent to make more believable decisions.

To provide this nuanced representation, information about the world must be prevented from being collected or stored "perfectly". Any fully digital behaviour simulation (as opposed to a behaviour system affecting an Agent in the real-world; a robot for example) is negatively affected by being perceived perfectly by the Agents, because with this perfect knowledge an Agent is able to make decisions that would otherwise be impossible to make without such knowledge. To prevent this from occurring, there are opportunities during the stages of world perception, memory storage, and memory retrieval to "corrupt" the information the Agent is trying to access. Information "corruption" at the perception stage has already been undertaken using a fuzzifier (Li et al., 2004), and as such I propose performing the "corruption" within the memory system of the Agent.

As discussed in section 4.2, memory loss can be a key aspect of human behaviour. By storing information about the world in a system that is subject to information loss, it is possible to generate behaviours that would otherwise have to be specifically applied to the Agent, rather than being dynamically generated by the Agent itself. An example of this would be the event of a person losing their house keys; this occurs when a memory retrieval failure causes the specific memory of the location of an object in the world to be irretrievable. In order to replicate this in an Agent without a biologically inspired system, we would have to specifically write a behaviour that has the Agent "lose" its keys, deleting that specific memory at a certain point in

its life cycle. It would be necessary to have the behaviour run only at realistic times (forgetting the keys immediately would be unrealistic), and to only be run a realistic amount (forgetting the keys every time would be unrealistic).

By aiming to store all aspects of the virtual world in a biologically inspired memory system, we are opening up the entire world representation to memory retrieval failure. This means that the Agent is capable of forgetting anything that is has learned, without a specific behaviour having to be encoded to accomplish this. This allows for the dynamic generation of behaviour while the Agent is online, without having to specifically consider all possible behaviours for every possible world aspect.

## 4.5 LIMITATIONS OF BIOLOGICAL MEMORY AND RELATED SIMULATIONS

In this section I examine some of the limitations studied in both the human memory system and my own methodology. I discuss various, sometimes conflicting, cognitive theories and consolidate these into a proposed methodology that forms what I believe is a reasonable compromise as well as allowing for its implementation in computational simulations.

### *4.5.1 CHANGE BLINDNESS*

The human visual system is subject to what could be considered a dramatic flaw known as change blindness (Rensink, 2001), simply put, the failure to detect changes in a visual scene when there is a transient interruption in the observation of the scene. Research has indicated that attention to an object in a scene is necessary to notice a change to that object (Rensink et al., 1997); this further indicates that humans never build a complete representation of the scene they are observing.

This issue within the human visual system elicits a difficult question when it comes to artificial agents; namely, should artificial agents be subject to change blindness? For the purposes of creating a semi-realistic simulation of human-like behaviour, it is desirable for an artificial agent to replicate biological systems as closely as possible, even when these systems appear to have limitations.

It is necessary to consider the fact that the human visual system appears to be the most effective method for analysing a scene, and that limitations such as change blindness exist for a reason; namely that it allows us to process large amounts of information with limited resources. Without change blindness our minds would be overwhelmed with the amount of information available to us in a visual presentation. This same issue can be reflected in computer simulation, where there is a finite amount of resources to be shared between many processes, not only to implement

various independent instances of agents, but also the simulation of the environment and its restrictions as a whole.

### *4.5.2 Limited Capacity of Short-Term Memory*

In comparison to Iconic memory, VSTM is relatively limited in its capacity of storage. Initial research has indicated that the capacity of STM is seven items, plus or minus two (Miller, 1956), however more recent discussion has lowered this capacity to four items (Cowan, 2001). However, the question of what constitutes an "item" remains, with some arguing that fully realised objects are items (Luck and Vogel, 1997)(Vogel et al., 2001)(Delvenne and Bruyer, 2004), and others discussing the theory that an object's features are items in their own right (Alvarez and Cavanagh, 2004)(Wheeler and Treisman, 2002).

At present there seems to be no conclusive evidence one way or the other, and as such, I investigated several methods as possibilities for storage in VSTM to form the consolidated proposed methodology.

### 4.5.3 Object Features as Individual Items

Figure 10 illustrates how an object can be decomposed into its individual features. As discussed in section 4.3.3 an objects' position is decoded and stored first, followed by other features.

Recent research has shown that test subjects are capable of remembering an increased number of lower-complexity objects rather than higher-complexity objects (Alvarez and Cavanagh, 2004). This directly contradicts previous research that has suggested a fixed amount of storage based solely on the amount of objects to be stored (Luck and Vogel, 1997). However in cases where only objects of minimum complexity are stored this does not mean that the total capacity of VSTM increases past the suggested limit of four items, rather only four to five objects can fit in storage and this limit does not increase for any stimulus type. Earlier research also supports the hypothesis of a trade-off between the number of objects stored and the detail of these objects (Palmer, 1990).

Given this knowledge concerning the capacity of VSTM, Figure 10 also illustrates a VSTM that is completely filled by one item. What then would occur if that one object had more attributes? Without extra storage space it would seem impossible to store these extra attributes. So would a test subject be unable to report them at all, or would repeated expose to the object elicit different responses?

While these questions have not been fully answered in relation to biological systems, it is still necessary to address them in regards to a technological implementation. In order to address these questions in the proposed system I have concluded that when VSTM becomes full, items in memory are overwritten based on when they were last accessed, thereby allowing memories to be rehearsed and retained even when there is a large amount of information to be processed. This solution also applies to the question of remembering a second object with the same amount of features; features that have not been recently accessed are replaced with features that the agent has just extracted.

### 4.5.4 Objects as Outside Memories

It has been suggested (O'Regan, 1992) that it is not necessary to maintain a representation of all objects within the immediate visual environment because if the objects are currently visible, they themselves can act as an "outside" memory.

It could be added, however, that there must be at least some minimal representation of each object. Otherwise how would we know which objects are available in our current view to examine further? The most unique feature of every object is its location, and as this information is of restricted complexity it is likely that this feature would take up only one "slot" in VSTM, however research has indicated that the amount of spatial locations does not influence VSTM capacity (Lee and Chun, 2001).

At some point in the cognitive process, it must be possible however to extract the features of visible objects and store them in memory as a recombined object, else it seems unlikely that we would be able to recall the layout of a room, where we left our keys, or some other visuospatial related task.

I have not directly implemented this methodology in the proposed system, however it was noticed that it exists implicitly to some extent in any virtual world that contains artificial agents;

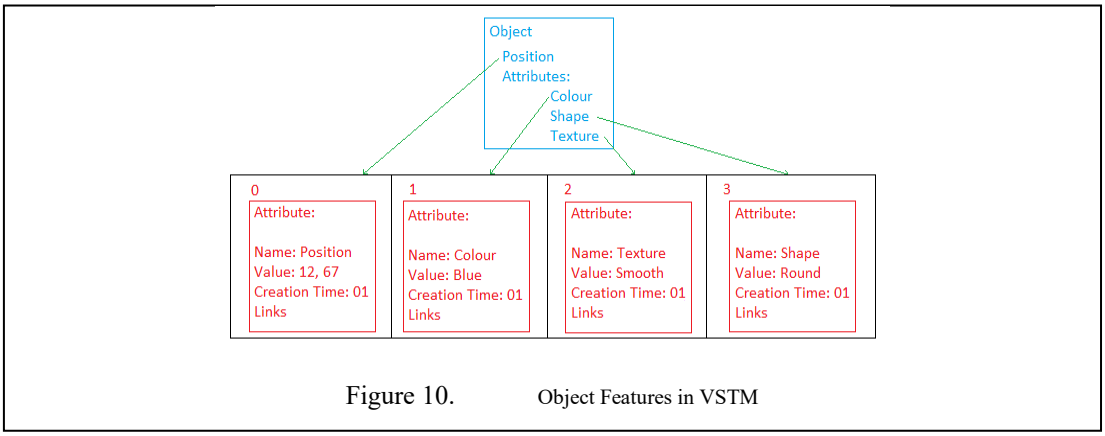


Figure 10. Object Features in VSTM

the world itself acts as the "outside" memory, providing a method for the agent to retrieve information about any visible object. If the agent remains at a given location, and observes the same scene or space for more than one cognitive processing frame, in a way it is using the environment itself as an "outside memory".

### 4.5.5 Unified Objects

Research indicates the possibility of objects being stored as a single unified object, as while only four features from one dimension can be recalled, further items from another feature dimension are also recallable (Luck and Vogel, 1997).

Figure 11 indicates how unified objects could be stored in VSTM. Each object has a set of features that are encoded into memory chunks of their own and then linked to a memory chunk for the object itself.

The most pressing issue with this theory is that it would seem to indicate that a seemingly unlimited amount of features could be stored for up to four objects, something that seems


Figure 11. Unified Objects in VSTM

unrealistic; four items with one feature must be simpler to store than four items with twenty features.

In terms of a technologically implemented system this is a very restrictive limitation, as the agents must have some form of upper limit to their memory capacity; we cannot therefore implement a system that allows an agent to store up to four items for an unlimited number of feature dimensions. For this reason, I have decided not to implement this methodology as part of the proposed system.

### 4.5.6 Independent Memory Stores

Figure 12 illustrates the concept of independent memory stores as described in recent research into the field (Delvenne and Bruyer, 2004)(Wheeler and Treisman, 2002). In (Delvenne and Bruyer, 2004), the authors suggest that each feature dimension (colour, orientation, shape, etc.) may have an independent VSTM store, with each store therefore having its own capacity. This suggestion is supported in other work, illustrating that memory for spatial frequencies and contrasts do not interfere with each other in a dual-memory task (Magnussen et al., 1996).

This theory would allow for the concurrent storage of four objects in a unified manner, yet still restrict the total storage of information across VSTM. Object features would still require encoding into memory chunks that are then linked to a unique object memory chunk to enable binding of features to each other.

This methodology seems to propose the most logical concept for storing objects in VSTM and as such I have drawn on this theory for the proposed system. As will be discussed further in section V, I have used this principle with some modifications to the capacity of the independent

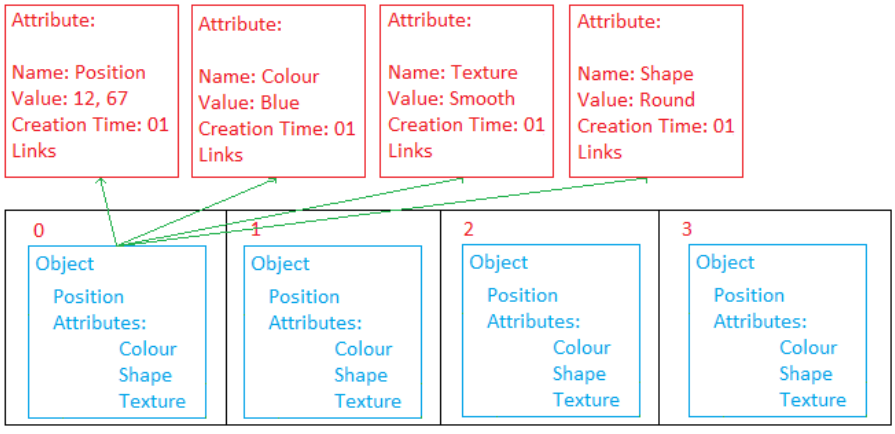


Figure 12. Independent Memory Stores

feature stores.

### 4.5.7 Overview of Limitations

With a vast range of conflicting research in the field of biological short-term memory it is difficult to unambiguously decide on which is the "correct" theory. I agree with (Alvarez and Cavanagh, 2004)(Wheeler and Treisman, 2002) that the capacity of VSTM is defined by two limits; namely that the total amount of visual information (features per object) cannot exceed the maximum information limit and the number of objects stored cannot exceed four or five. Storage space for an object's features must come from somewhere, and if VSTM is limited to four items then we cannot logically store four objects as unified items.

I have not explicitly defined a method to implement the concept of objects acting as external memories as proposed in (O'Regan, 1992), however it is interesting to note that in an artificial simulation, this concept could actually be considered to exist implicitly; the virtual world itself acting as the external memory, the agent being able to query the virtual world by way of its visual processing system.

# CHAPTER 5. PROPOSED SYSTEM

## 5. INTRODUCTION

In order to hybridise the behaviour methodologies that I have investigated, along with the biologically inspired memory system that I have discussed, it is necessary to make some adjustments to these existing methodologies so that they better complement each other. These methodologies were not originally designed to be part of a hybrid system, so it is understandable that adjustments are necessary for each individual component.

In this chapter I cover the adjustments I made to the Behaviour System, as well as the Belief Network that was previously discussed in Chapter 4. The hybrid Behaviour System is, by its nature, split into both a Finite State Machine and Belief Desire Intention system, and I discuss these methodologies separately.

The Belief Network necessitated the most adjustment as it had previously not been implemented in a working test model and was therefore mostly theoretical before implementation began. As the implementation process continued it became clear that more significant alterations were necessary to construct a system that was workable at real-time speeds.

## 5.1 BEHAVIOUR SYSTEM

The different behaviour methodologies investigated previously require some degree of modification in order to hybridise them effectively.

These modifications are minimal and most commonly involve the complexities of implementing behaviour systems that can cope with varying degrees of uncertainty. In the case of Finite State Machines we must provide a simple method for defining States, as well as reducing the number of possible states to a minimum. For Belief Desire Intention systems we must provide a method for the Agent to examine its surroundings, update its Belief network, and maintain a Belief network that itself behaves in a realistic manner.

### *5.1.1 FINITE STATE MACHINE*

A Finite State Machine commonly would only transition between states that are entirely self-contained, however in a hybrid system an entire state is turned over to the Belief Desire

Intention system. This results in a FSM that spends the majority of its time in a single state that does not affect the FSM itself.

In my hybrid model, the purpose of the FSM is to emulate biological-like Instincts, providing a method to quickly react to an external stimulus in such a way that the Instinct is resolved in a manner positive to the Agent. In order to realistically accomplish this, it is necessary to reduce the number of possible States and Transitions that the FSM can enter into an absolute minimum, and ensure that after the State has been resolved the FSM returns control to the BDI system, rather than flipping between other States.

To allow for additional Instincts to be added, a simple definition file can be utilised. The file specifies the external stimulus required to activate the State, and the Action to take to resolve the Instinct.

```
<instinct name="escapeFire">
    <trigger>
        <nearTo>
            <object name=”fire” />
        </nearTo>
    </trigger>
    <resolution>
        <notNearTo>
            <object name=”fire” />
        </notNearTo>
    </resolution>
</instinct>
```

Figure 13. Example Instinct Definition

### *5.1.2 Belief Desire Intention System*

While Belief Desire Intention systems have been implemented before (Braubach et al., 2005) (Huber, 1999), these implementations do not appear to be particularly extensible in regard to offloading some of their processing to another behaviour system. For this reason I found that it would be more practical to re-define the BDI system rather than trying to alter an existing implementation beyond its original design specifications.

The construction of a BDI system raises its own challenges, particularly when deciding on the most appropriate method to use for the description of Objects in the virtual world and their Attributes, along with a method for describing Desires and Plans for the Agent, as well as Attributes of the Agents themselves.

In order to simplify the description of all properties in the virtual world, simple XML files are used as they offer simple flexibility and extensibility. Similar properties are described in different definitions using the same tag names, making them easy to identify and match up.

#### 5.1.2.1 ATTRIBUTES

Attributes are described simply as having a Type, Name and Value.

The name does not affect the system and is used for human identification, but it is used for comparison with other attributes and as such it should match up when later used in Plans or Desires. It is not case sensitive.

The value of an attribute is specified as a string, its type being coerced from the attributes' type when the attribute is read by the system at run-time.

```
<attribute type="integer">
    <name>hunger</name>
    <value>0</value>
</attribute>
```

Figure 14. Example Attribute Definition

#### 5.1.2.2 DESIRES

Desires consist of a "starting" state and a "target" state. The "starting" state is the state in which the Agent must be in order for the Desire to be considered a valid desire at the current moment. The "target" state indicates the state that the Agent will be in once the Desire has been

```
<desire name="eat" priority="1">
    <startingState>
        <attributes>
            <attribute type="integer">
                <name>hunger</name>
                <operator>GREATER_THAN</operator>
                <value>50</value>
            </attribute>
        </attributes>
    </startingState>
    <targetState>
        <attributes>
            <attribute type="integer">
                <name>hunger</name>
                <operator>LESS_THAN</operator>
                <value>50</value>
            </attribute>
        </attributes>
    </targetState>
</desire>
```

Figure 15. Example Desire Definition

resolved. By examining the "starting" and "target" states and comparing the Attributes of the Agent that are affected by the Desire, it is possible to calculate the "Consequence" of the Desire being fulfilled.

A desire also has a name and a priority. The priority is considered when comparing which valid Desires should be acted upon. Attributes in a Desire's "starting" or "target" state are defined as per normal Attributes, but have an extra field called "operator". This value can be one of several predefined options, which indicate a mathematical operation that should be performed on the value of the Attribute when it is evaluated.

#### 5.1.2.3 PLANS

Plans are defined by a list of Consequences that are achieved by executing the Plan, a list of Requirements that must be in place before the Plan can be executed, and a list of Actions that define what the plan actually does.

There are several predefined types that a Requirement can have; either "haveItem", "seeItem", or "beAt". These Requirement types describe a situation that must be fulfilled in order for a Plan to be deemed as executable. If the Requirements are not met, a Plan cannot be executed.

Each Consequence in a Plan has an ID, which is matched by a "fulfils" attribute on each Action. This allows us to apply the Consequence even if the Action does not directly cause that particular consequence.

```
<plan name="pick up food">
    <consequences>
        <consequence id="1" type="item">
            <getItem>
                <class>food</class>
            </getItem>
        </consequence>
    </consequences>
    <requirements>
        <requirement>
            <seeItem>
                <class>food</class>
            </seeItem>
        </requirement>
    </requirements>
    <actions>
        <action fulfils="1">
            <getItem>
                <objectSelector type="class">food</objectSelector>
            </getItem>
        </action>
    </actions>
</plan>
```

Figure 16. Example Plan Definition

Every Action is one of several predefined types, which include; “getItem”, “seeItem”, and “beAt”. In the case of the “beAt” action, further information is required in the form of a Location tag. This tag is made up of an “x” and “y” coordinate that defines a specific location in a 2D space that indicates where the Agent should “beAt”. For “getItem” and “seeItem” actions, it is necessary to specify the Item you are trying to affect. This is achievable using the “objectSelector” tag, which provides a method to describe the Item you wish the Action to affect. An “objectSelector” can define either an Object’s class, or its name.

## 5.2 Belief Network

To achieve a more realistic simulation of behaviour, an Agent must be able to store information in its belief network, and expect to be able to retrieve this information at a future time, with the possibility of failure, especially if the memory has not been recalled recently.

The ability to store and recall huge amounts of information is something that we as humans take for granted. While we cannot always remember something perfectly, or the first time we try, we are commonly able to recall information when we are given some form of cue (Bjork and Bjork, 1992), or if we try to remember for long enough.

For the purposes of Serious Games and Artificial Life Simulations, Intelligent Agents commonly have an unfair advantage over their real-life counterparts; namely that they never forget. When applying a generic method of degradation, this can lead to memories that are completely lost, whereas in biological memory these memories are not lost, rather they become non-recallable with disuse (Estes, 1997).

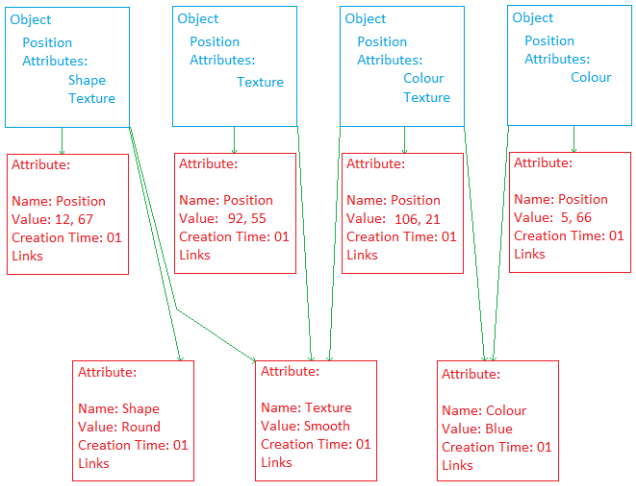


Figure 17. Proposed VSTM layout

By examining research in the field of biological memory systems, I have been able to form an artificial memory system that emulates basic human memory concepts. Research is still inconclusive as to the exact nature of biological memory and how objects are processed and stored but I have been able to build an approximation based on the information that is available.

The proposed system implements the concept of independent memory stores for individual features of an object. There are few differences between each feature dimension having its own memory store and an object's features taking up one slot in VSTM. With the former, up to four objects could have up to four features recalled perfectly, the latter, up to four objects could have one feature recalled perfectly. I believe some median between these two could be found; perfect recall of sixteen individual features appears to be excessive. The question of whether features with the same value take up one memory item or multiple memory items remains, and the resolution of this question could prove most informative.

Consider four objects with identical colour, but other features that differ entirely. If all feature dimensions have their own memory store, and each value is stored regardless of duplicates, this would require sixteen memory slots, whereas removing duplicates and having multiple objects reference the same memory item would reduce this memory load to thirteen slots. It seems unlikely that a system as streamlined as human memory would create unnecessary duplicates.

I have resolved this issue in this implementation by having one memory item per unique feature. Any object that has a feature with a matching value then links to this memory item. This also allows us to search for objects in a reverse manner; a search for objects linking to the "blue" memory item would return all the objects that have "blue" as one of their features. Figure 17 shows four unique objects that share some of their attributes with each other.

Our proposed system consists of four objects each with its own unique associated position. There are also up to four different features, which can have any combination of links between the object chunks. The number of possible permutations of links between object chunks and features is finite. In addition, in this structure, this number is relatively low, so it is possible to implement this system using a limited number of memory resources. Figure 17 shows a case where one additional feature could be stored and some specific links have been made.

### *5.2.1 Dynamic Memory Recall*

I aim to provide a method of storing information in such a way that it can be recalled with varying degrees of success, however ensuring that data is never lost once it has been stored in long-term memory. I provide systems for memory rehearsal and recall using an abstract data type known as a Chunk in order to remain non-specific to the types of data the system is able to

store. Chunks can only be recalled from long-term memory by following a link, which means the system has no direct way of information recall, thereby negating the need to store a map, list or array of memories.

It should be noted that in this work we are not seeking to use the widely researched methods for using associative memory as a means to achieve efficient content indexing alternatives (Morelli et al., 2004)(Bohland and Minai, 2001). Instead, I am using the approach inspired by generic use of associative memory (Murdock, 1982) and biological systems to create links and paths to make it possible to reach all memory content in the LTM regardless of the type of association it has.

### 5.2.2 Adjusted Model

In order to utilise the Multi-Store model, it must first be added to in order to fit an artificial paradigm. These additions should not alter the core theory of the Multi-Store model in any way, but rather simplify how it can be implemented in an Artificial Life scenario.

#### 5.2.2.1 “Chunked” Memory

While biological memory stores different types of information in different ways, for simplicity we will store all types of information in the same way, henceforth known as a Chunk. A chunk has no particular data type itself, but contains a pointer to a section of data. The chunk also contains a list of links to other chunks, and a list of their data types respectively.

#### 5.2.2.2 "Linked" Memory

I aim to be able to achieve a method of cued memory; by this I mean the ability to recall chunks when given an external or internal cue. To this end we must define memory as being linked together in some fashion, whether this be based on a common aspect of the memories being linked or merely their “closeness” in storage (time based). When these links are formed they are given a strength, which ranges between 0 and 1. This strength is considered when attempting to retrieve a chunk that the link points to. Once created, links cannot be unmade, and there can be multiple links per chunk.

#### 5.2.2.3 Store Conversion

It is necessary to consider a method for moving chunks from one store to another; most commonly in moving from STM to LTM. It has been suggested that in biological memory, information is transferred into LTM from STM and that this amount and form is determined by the control processes in the latter (Atkinson and Shiffrin, 1976). As we are dealing purely in an

artificial implementation, the luxury of this biological process does not exist, however it is possible to force the system to use some form of chunk rehearsal counter to simulate this.

We will define that any chunks entering the SM store will be held for a time as in a biological system, and will only enter STM if directly accessed. Any chunk that is not accessed within this time frame is lost, and the SM store has a maximum amount of chunks it can hold, with any extra chunks overwriting existing ones (oldest first).

The STM store retains chunks for up to one minute before they are permanently deleted. During this time, if a chunk is reconsidered (rehearsed), its deletion timer is reset to 0. The STM store also has a maximum capacity, and any extra chunks entering will overwrite existing ones, with the chunks closest to deletion being overwritten first. Once a chunk has been rehearsed 5 times it is copied to the LTM store.

The LTM store retains chunks indefinitely. When a chunk is to be stored in LTM it must be associated (linked) with at least one other chunk, or it is lost, however a chunk can have multiple chunks linked to it. When a chunk is moved from LTM back to STM for consideration, it is effectively copied; the original is still in LTM and any changes made while in STM must be copied back to the LTM chunk.

### 5.2.3 Storing Memory

Storing memory chunks in an appropriate manner is key to the viability of this system. If memory chunks are stored incorrectly they could become completely inaccessible, thereby becoming (in programming terms) a memory leak. With the amount of data the system should be capable of handling, stored yet inaccessible memory is unacceptable.

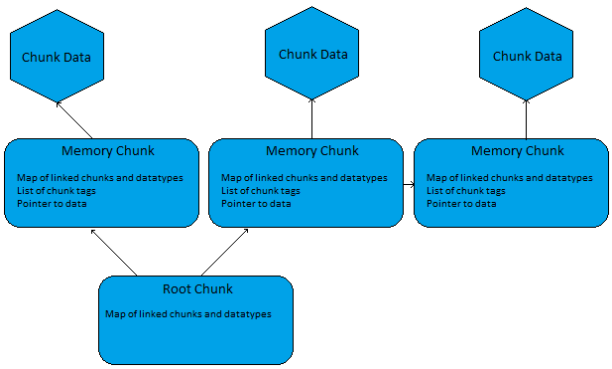


Figure 18. Memory Chunk Layout

To address this problem I have specified that when a memory chunk is stored, it must be linked with an existing chunk. In order for this to work with the first new memory chunk stored, we must define a root memory chunk. It is only necessary to do this in the LTM store, as both SM and STM stores have a finite size and it is possible to access their contents directly. The layout of chunks in memory is detailed in Fig. 2.

Our root chunk is identical to other memory chunks in that it contains a list of chunks that it links to and a corresponding list of their data types, however its pointer to data is set to null. This chunk can be seen as our starting point in memory.

When we wish to store our first chunk of memory, we create a link between the root chunk and our new chunk. We then tag our latest chunk with identifiers. Subsequent chunks to be stored will access the Last Chunk Stored (LCS) from the STM store and search for matching identifiers to a depth of 2 child chunks. STM will also be searched for any relevant chunks to link to.

#### 5.2.3.1 Chunk identifiers

A chunk identifier is a piece of information that we can use in order to search our memory chunks. Without such information it would be necessary to scan, interpret and search the contents of every chunk in order to find chunk similarities for the purpose of linking the chunks together.

At least one chunk identifier should be added to every chunk in memory to facilitate the linking of chunks. The more identifiers a chunk has, the more chunks we can potentially link it to, and searching for the chunk becomes potentially faster.

All chunk identifiers should be of the same type for simple interpretation. In this implementation I suggest using the std::string type in C++, however other types could be used, for example a flag system using integers, or a char array.

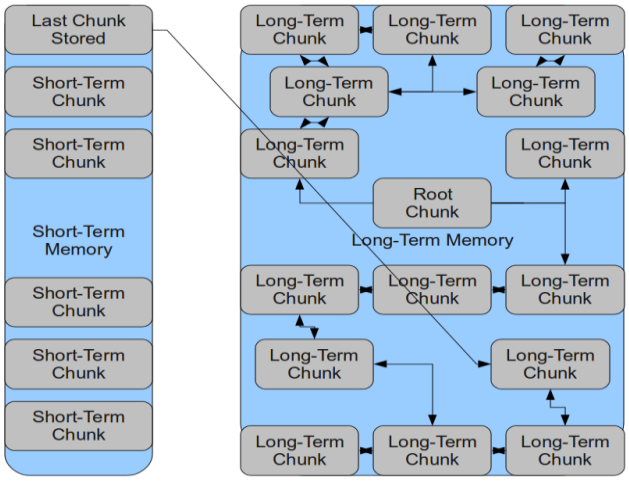


Figure 19. Last Chunk Stored

5.2.3.2 Last Chunk Stored

For the purposes of linking memory chunks quickly without the need for an extensive search throughout LTM, a way of accessing the most recently stored chunks is required. To this end I suggest implementing "Last Chunk Stored". This is a permanent slot in STM that points to the last memory chunk added to LTM. This can be thought of as the "thing you were last thinking about"; a quick access to your most recent thoughts.

LCS is updated every time a new chunk is moved from STM to LTM. The value is never null except before the first memory chunk is stored. LCS is illustrated in Fig. 3.

5.2.3.3 Background Linking

To keep processing times down when storing new memory chunks in LTM, it is necessary to perform only a cursory search of related chunks to link to. However this does not provide the most effective method of later searching for memory chunks, and as such, memory chunks must be better organised at a more convenient time. To address this, I have designed a system known as Background Linking (BL).

Research in the field (Roth et al., 2001) argues both for and against the possibility that sleep is the period in which memory is organised in humans. While not definitely proven either way, I have found it useful to provide sleep as an example when describing BL. It has been indicated that deprivation of REM sleep in individuals can result in drastically lessened capacity to retain

memory from previous activities, and this relates strongly to what I am describing; without BL an Intelligent Agent has much less chance of successfully recalling a memory chunk.

When an Agent has spare processing time, for example when it is not engaged in any other task, it should hand over its spare processing time to our memory system. This will then begin a search throughout LTM for any chunks that have a relatively small amount of links. Chunks that have only recently been stored and have not been subjected to a BL process before are likely to only have one link. During this BL cycle, poorly-linked chunks will be analysed for chunk identifiers, and then these identifiers will be searched for elsewhere in LTM. Any matching chunks will then be linked up.

5.2.3.4 Link Strength

Link strength is an important component of memory recall, however for it to be used in this system it must first be set up during memory storage. Link strength symbolises how regular a path between memory chunks is followed. This value ranges between 0 and 1 and can either increase or decrease based on the usage of the path between memories. Link strength is similar to the "retrieval strength" described in (Bjork and Bjork, 1992).

When a memory chunk is linked to, or linked from, the link strength should be a value of 1. This indicates that the link is in perfect condition. In a biological system, it is likely that a chemical change over time lessens any "retrieval strength", however in this artificial system there is no such method given to us automatically. For this reason it is necessary to store also a time stamp indicating when a link was last accessed. When the link is next accessed this time stamp should be considered, and the link strength be altered accordingly, with a longer period of disuse resulting in a larger decrement to link strength. Classically the nature of forgetting has been described (Ebbinghaus et al., 1913) as in (1), where R is memory retention, S is the relative strength of memory and t is time.

$$R = e^{-\frac{t}{S}} \quad 1)$$

We modify this Forgetting Curve when determining the link strength as described in (2), where "S" is the new link strength, "t" is time since the link was last accessed, and "c" is the current link strength. This curve differs from the classical representation because it feeds back into itself. Where the classic representation suggests that the chance of memory retention is evaluated once, based upon the strength of that memory, we instead evaluate the chance of

accessing a memory each time we attempt to do so, with the result of the attempt to access the memory affecting attempts in the future.

$$S = e - \frac{t}{c} \quad 2)$$

In addition to the creation of a link strength, all other links to or from the new chunk should be lessened slightly. Consider when a person moves house; when asked about their current address it is likely they may initially make a mistake and repeat their previous address, whereas with time and increased access of the new address the ability to recall the old address lessens somewhat.

### 5.2.4 Retrieving Memory

With memory chunks stored, we must define a method in which these chunks can then be recalled or retrieved from their stores. As we have been inspired by biological systems, we want to ensure that this system is capable of failure, yet also capable of fast retrieval of more recently stored chunks.

#### 5.2.4.1 Chunk Searching

Searching for the correct chunk of memory to return is a key aspect of memory recall. Chunk searching achieves this by moving through the LTM store, checking each chunk for its identifiers, and matching these against the search parameters. One matching chunk identifier is enough for a positive result; however this can lead to situations where an incorrect result is returned. In a normal AI application this would be considered a failure, but in this simulation of human memory this is actually accurate.

Research has indicated that the process of searching the human memory can become fixated on an unsuccessful search procedure (Atkinson and Shiffrin, 1976); for example when we are aware that we know the answer to the question, but cannot recall the answer at that time – usually when we begin to think of something else, the answer comes to us. This shows that abandoning a running search is an important aspect when dealing with such a large set of data as is contained within human memory.

Multiple chunk searches can be run simultaneously; however this could cause a severe reduction in performance. Running a single search from multiple chunks is a more appropriate method, allowing us to start from either the Root Chunk in LTM, or the LCS.

Chunk searching should also take into account any chunks currently stored in the STM store that are copies of chunks in the LTM store (these can be identified by having links to other chunks; new chunks that have not yet been stored in LTM have no links). By searching from these chunks as starting points it is possible to quickly access a location in the LTM store that has recently been used, something that is likely to assist us during an involved search process. Providing a context reduces the amount of computation needed to perform a search that is most likely to return results related to the current problem as memory chunks within the context are the most likely chunks to be involved in the current actions being performed by the Agent, or the actions it has most recently performed.

When a chunk has been found that matches the search parameters, it should be copied out to the STM store before being returned. This allows near-future searches to access this chunk more quickly, rather than having to repeat the entire search from scratch.

### 5.2.4.2 Link Strength

During the chunk searching process, we must take into account the link strength between memory chunks. This link strength has been designed to indicate how often a search path through memory is followed, somewhat similar to ant pheromone trails resulting in the fastest path between a food source and home (Parunak, 1997). As the link strength ranges between 0 and 1, we should follow the strongest links between memory chunks; however we should "touch" every link in order to increase its strength with repeating searching. We must

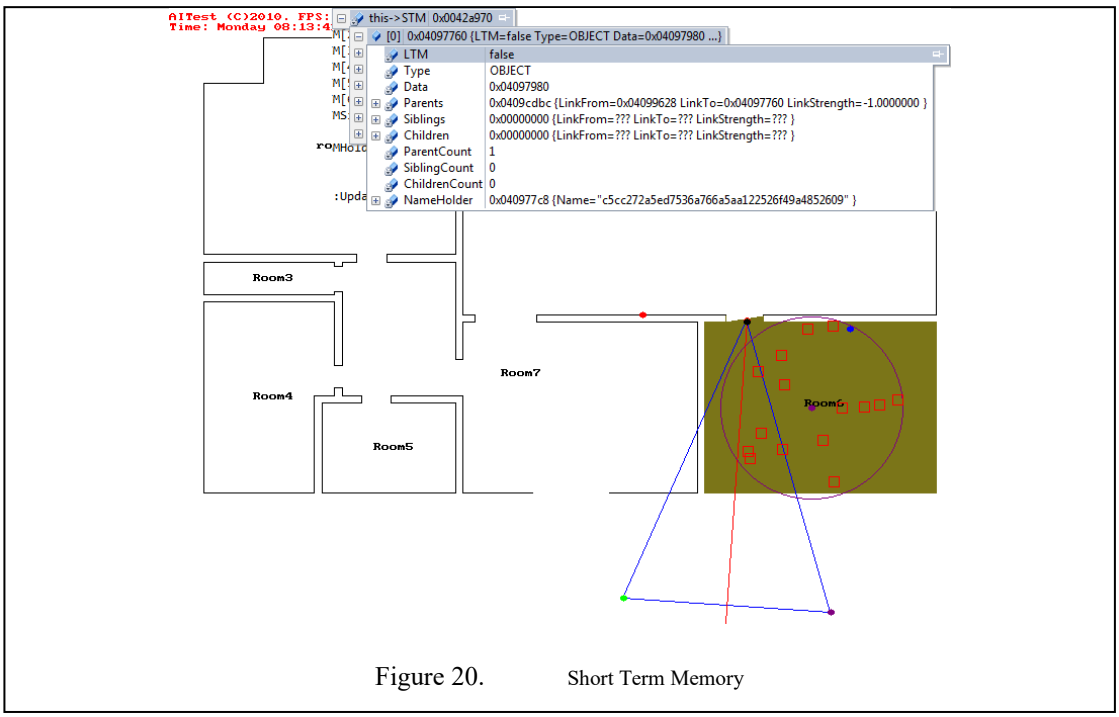


Figure 20. Short Term Memory

also check the time stamp associated with each link to determine if the link strength should be decreased due to lack of use. If we encounter a situation where multiple links have the same strength (which is stronger than all other links), the link with the oldest time stamp should be followed. This allows us to break a deadlock over chunks with similar link strengths. If one search follows a path and results in an incorrect result, further searches need to follow an alternate path to prevent the same result being returned. Once a link has been followed its time stamp should be update to indicate this.

#### 5.2.4.3 Diminishing Search Strength

As suggested in (Bjork and Bjork, 1992), it is a necessity that we lose retrieval access to information that we no longer use; in both biological and technological systems alike a search cannot continue forever, and (Atkinson and Shiffrin, 1976) suggest that "one of the decisions a subject must make is when to terminate an unsuccessful search".

Our system of link strength facilitates this requirement, as any links with a low strength are not followed in preference to links with a higher strength. This effectively limits the amount of search options available to an Agent at any given time, only increasing temporarily when a search is attempted.

To provide another method of limiting the amount of time spent on a search, I also propose a system of diminishing search strength. The proposed system requires that any search begins with a strength of 1, and every link that the search follows reduces the search strength by a given amount, for example 0.1, thereby giving the search a maximal amount of links that it can pass through before being abandoned when search strength reaches 0.

If a search fails or is abandoned and is attempted again, further options have now been opened in the search via the "touching" of links described in the previous section.

#### 5.2.4.4 Short-Term Search

The STM store, while limited in capacity should also be searched when attempting to retrieve a memory. As each chunk in the STM store is listed directly, it is a simple case of iterating over each chunk testing for chunk identifiers. During a short-term search links between chunks should not be followed or updated as this is handled as part of the LTM store chunk searching.

Figure 20 shows an Agent observing objects in a virtual world, with one of these objects being stored in the Agent's STM. Each of the STM slots is available for searching at this time.

### 5.2.5 Localised Memory

In order to maintain a rapid method of searching an Agent's memory space, it is necessary to reduce the actual search space that the Agent must traverse in order to do so. By localising memory chunks so that a memory pointer is updated as the Agent moves around, we are able to quickly adapt the search space to more accurately reflect what the Agent should be considering at any given time.

As an Agent moves around a virtual space, it quickly accumulates memory chunks for the objects that it encounters. As discussed, these memory chunks contain links to other memory chunks, thereby building up a "network" of chunks that accurately describe an "area" that the Agent has visited. As the amount of chunks expands however, so too does the amount of processing an Agent must perform in order to find any of these chunks again in the future. If an Agent were to always begin its chunk searches in the same "starting" location, it would have to search through an ever increasing linked-list to successfully iterate the entire memory space.

To combat this ever-increasing search space, an Agent must keep track of where it currently is in regards to its mental "network". A simple approach to this is by grouping memory chunks that share some common features (in this case, physical location), and regularly updating a memory chunk that points (links) to this grouping. By splitting the virtual world into a series of unique convex polygons, we are able to determine when an Agent has traversed a polygon boundary, and therefore when we need to update the localisation memory chunk pointer. As the

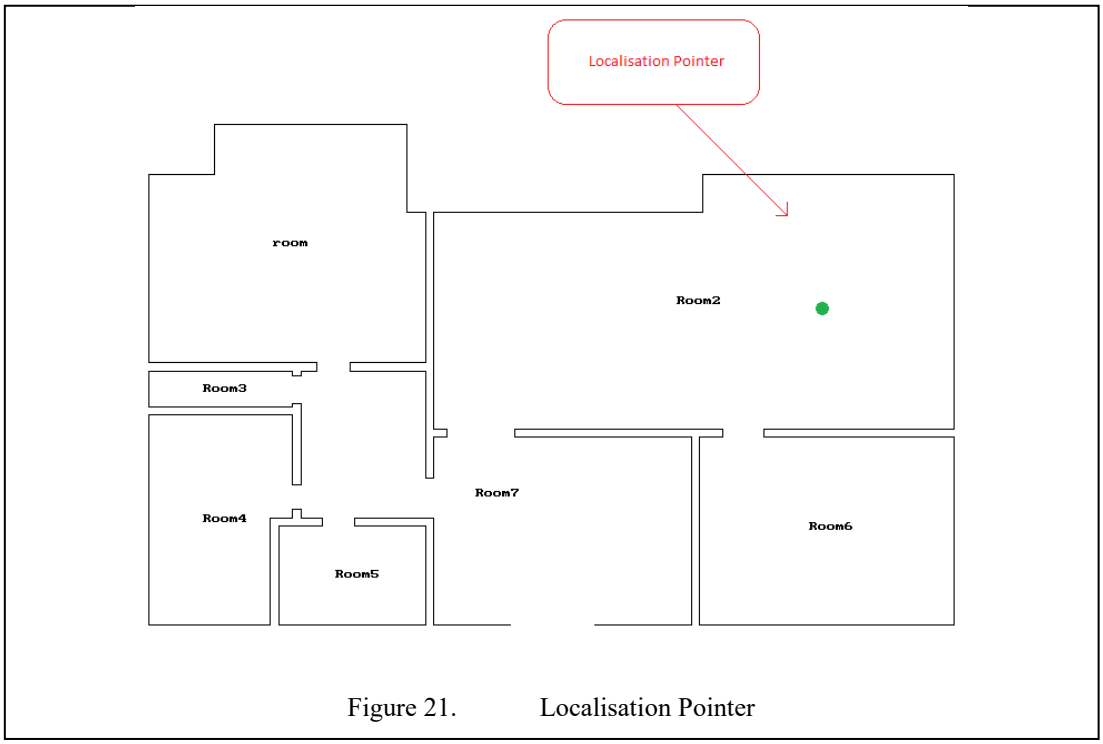


Figure 21. Localisation Pointer

Agent moves through the virtual world, it searches through all available localisation pointers to find one that matches its current local polygon.

With this grouping method however comes the added problem that the amount of group chunks will also steadily increase as the Agent explores more of the virtual world (although at a slower rate to that of object chunks). This has essentially delayed the problem but not avoided it. To ensure that an Agent is always able to retrieve its localisation pointer with a minimal search time, a further layer must be added.

By giving each localisation pointer a unique attribute, it is possible to identify each pointer individually. To achieve this, a SHA1 hash of the points that make up the Agent's current local polygon is taken, and applied as a unique attribute of the localisation pointer. This reduces the complexity of searching for the pointer, resulting in a simple string comparison of the SHA1 hashes.

In addition to this, as the localisation pointer is simply a memory chunk, we are able to link the pointer in a number of ways, namely to siblings (other localisation pointers), and children (the localised group). By taking advantage of this, we automatically link any new localisation pointer that is created (when an Agent encounters a polygon it has not entered before) with the

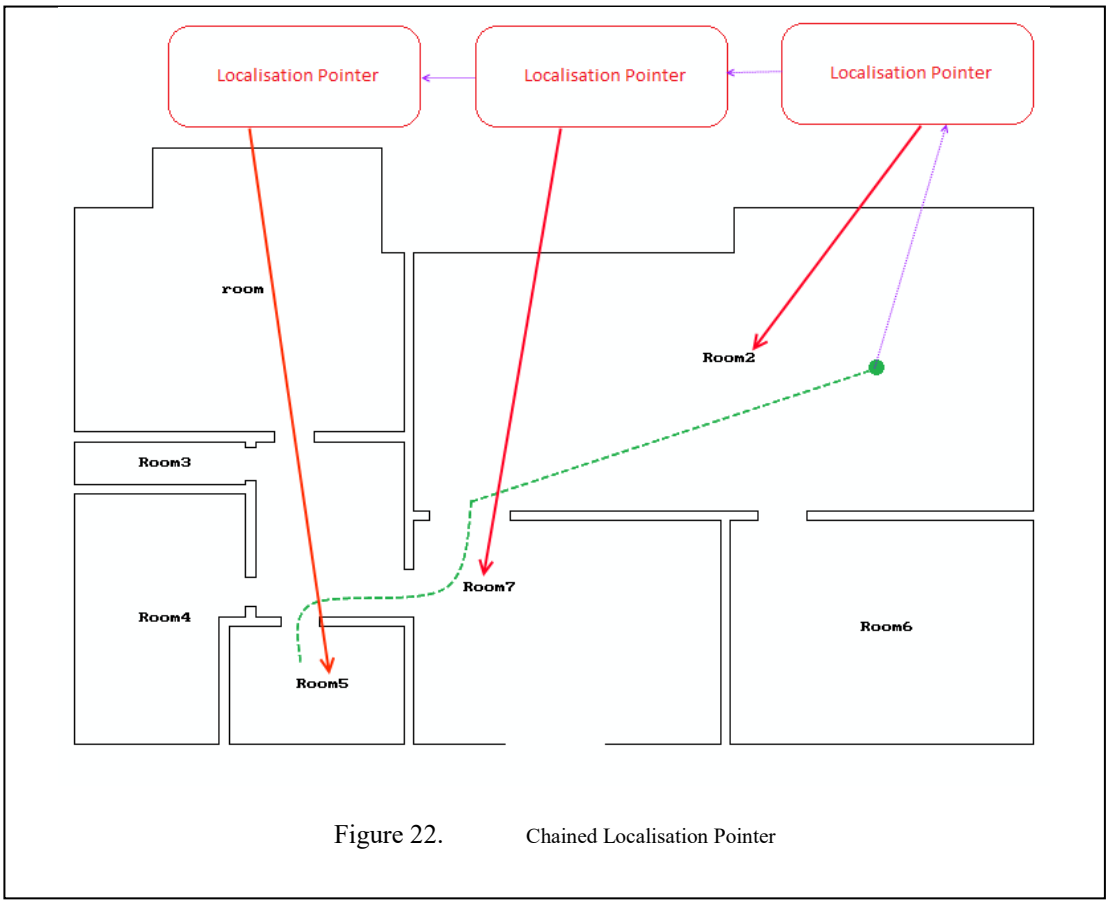


Figure 22. Chained Localisation Pointer

previous localisation pointer, thereby creating a linked-list of these pointers.

By using this linked list, the Agent is always aware of its most recent localisation pointer, and therefore of the previous localisation pointers in the event that it backtracks into a polygon it has previously visited. Given enough time to fully explore the virtual environment, the Agent will eventually construct a complete linked-list that fully describes such environment, allowing it to quickly access any of the localised pointers with either a 1 or 2 stage search.

Figure 22 shows an Agent with a movement history from “Room5” to “Room7” and finally to “Room2”. As the Agent has progressed from room to room it has created Localisation Pointers which refer to the rooms it has passed through. As each Localisation Pointer is created it is linked to the previous Localisation Pointer, allowing the Agent to follow the chain of pointers back through its memory to reconstruct its path through the environment.

## 5.3 Full Agent Structure

While the separate behaviour systems (Finite State Machine, BDI System) of a hybrid Agent could function by themselves, it is necessary to combine the systems as a whole in order to achieve the most effective methodology. The hybrid system described in this section has been designed so that each individual system complements the others in such a way that removing any of the individual systems would result in an overall methodology that is not as effective in producing semi-realistic behaviour.

The systems integrate with each other as follows:

- The FSM behaviour system provides quick reactions to situations, but its range of behaviour is limited.
- The BDI behaviour system allows for a wider range of behaviour, but its reaction time is significantly longer than the FSM system.
- The biologically inspired memory system provides a world representation for the FSM and BDI systems, while also providing the opportunity to generate unique behaviour by recalling information in a manner similar to that found in biological entities.

A possible future expansion of the system (discussed in more depth in Chapter 7 – Further Work) would be the fuzzification of the Perception system built into the Agents. By reducing the reliability of input from the Virtual World, and the confidence that the Agent has that such

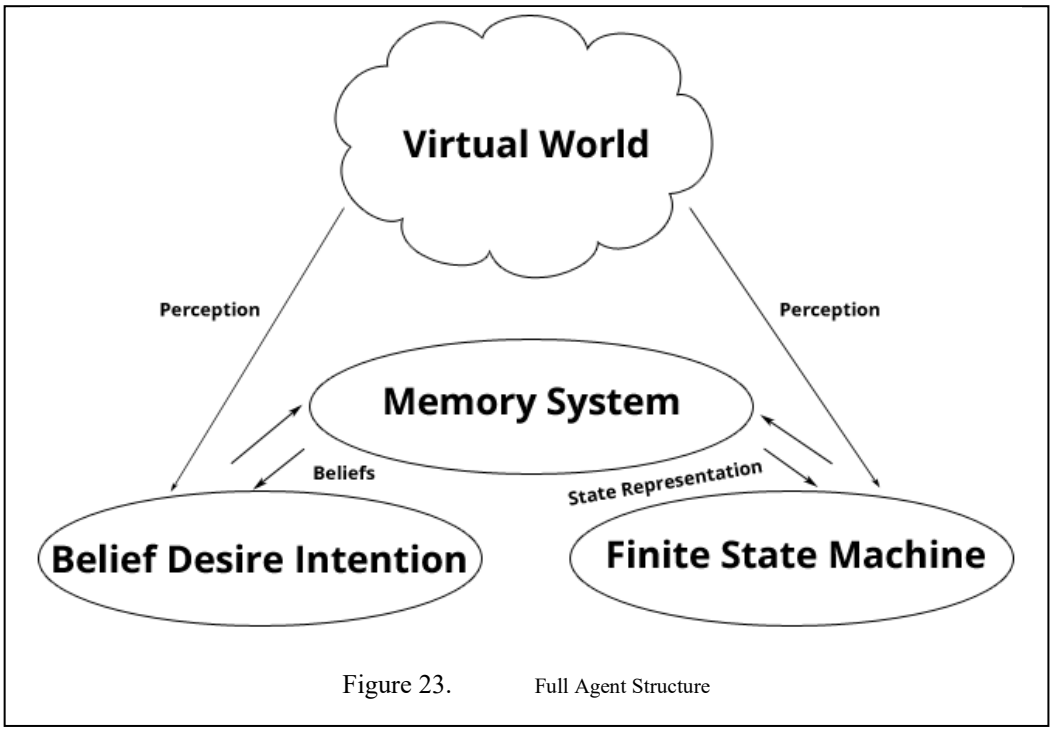


Figure 23. Full Agent Structure

input is reliable, further opportunities for unique behaviour present themselves, such as an Agent moving closer to a visual target to be more certain of what it is seeing, or investigating unknown sources of input for further clarification.

The Agents described in this section are considerably more scalable than either a Finite State Machine or BDI based system would be individually. By separating out behaviours into different categories of behaviour (Instinct vs. Learned) it is possible to minimise the amount of behaviours that need to be concurrently handled by either system, thereby increasing the upper limit of behaviours that can effectively be executed together.

These Agents are further capable of a form of learning, whereby a Desire that the Agent has been assigned could be completed in a number of ways. As the Agent develops more Beliefs (Knowledge) about the world it is able to choose the most effective Plan to achieve a Desire, as well as forming new Plans based upon its most recent Beliefs. For example this could take the form of an Agent getting a job that is closer to their home than their previous job, as this would be a more efficient use of their time (less time in transit).

# CHAPTER 6. CONCLUSION

## 6. CONCLUSION

By examining how individual behaviour systems contain natural limitations in their ability to provide realistic behaviour, I was able to suggest that a combination or hybrid of these systems can result in more realistic behaviour than was possible with the systems as separate methodologies. This hybridisation of systems allows for each system to complement the other, filling in gaps in the methodologies themselves, and has also allowed for the creation of Agents that can evolve their own behaviours (Gongora and Irvine, 2010a).

Through testing methodologies to employ as the Belief network in a Belief Desire Intention system as part of the hybrid Agent it became necessary to investigate more thoroughly the role that memory plays in generating behaviour. While it appears that biologically inspired memory systems have been implemented previously (Skubic et al., 2004), they have not been used in the generation of human like behaviour. By considering the proposition that the properties of a biological memory system directly contribute to the behaviour exhibited by humans, and enables them to model the world which they inhabit, I suggested that emulating a biological memory system would result in more realistic behaviour in artificial agents.

The process of implementing a biologically inspired memory system revealed other considerations relating to behaviour that are influenced by a memory system that has inherent imperfections, such as a contextual memory failure due to spatial change (Radvansky et al., 2011), or behaviour that comes from "tip-of-the-tongue" moments, such as repeated attempts at memory recall followed by eventual search abandonment. In addition to these considerations of behaviour, it also became clear that further enhancements to my original concept of the system were necessary, such as the necessity to store a "Localised memory pointer" in order to reduce the amount of time taken to search an Agent's belief network.

Constructing test bed systems for both the Belief Desire Intention and Finite State Machine behaviour methodologies has raised some further limitations with these individual methodologies that I had not previously considered.

A primary consideration of the work was to reduce the difficulty in prioritising behaviours within the Belief Desire Intention system, and the method selected for this was to extract high

priority behaviours and handle them inside the Finite State Machine. It became clear however that the behaviours that remained in the BDI system still needed to be prioritised in some way, as States where it would be suitable to select multiple Desires were still occurring, so to combat this Desires were given priority values, which therefore resulted in it being difficult to prioritise behaviours while the Agents were online.

Hybrid Agents offer considerable promise as a methodology for creating semi-realistic behaviour in situations where computation time is a factor and code complexity should be kept to a minimum. It has been shown (Bartish and Thevathayan, 2002) that Finite State Machines alone result in highly complex code as the amount of possible states increase, but that Belief Desire Intention Agents require more computation time when the same situation is true. It continues to appear that state of the art behaviour implementations rely on a "silver bullet" approach whereby one methodology is used by itself (Yue and Byl, 2006), rather than attempting to overcome the limitations of one methodology with the strengths of another.

Behaviour systems that aim to provide realistic representations of human behaviour should entertain the concept that human behaviour is governed as much by the apparent flaws in human biological processes as by the inherent complexity in such processes. To attempt to create a representation of a human that is capable of forgetting what it has done and things it has seen, it is first necessary to implement systems that enable this sort of behaviour; systems that are capable of forgetting. With my novel proposal I have indicated that it is simpler to develop a system that is capable of forgetting, than having a system with perfect recall and attempting to simulate this system forgetting certain things, as any form of pattern to such forgetfulness is easily recognised by humans.

The novel methodology that I propose for creating hybrid behaviour systems enables the creation of "future proof" Agents, that is, Agents that will remain relevant as computing power increases, and more advanced computing systems become more widely available. This acts as a first step towards the long term goal of Agents that behave realistically. While I have selected only two behaviour systems, BDI and FSM for my study, my proposed method is suitable for scaling up to additional behaviour systems, or behaviour systems that are designed after this work. By selecting specific types of behaviour to be handled by the different behaviour systems, any number of concurrent behaviour systems could theoretically be supported, the key aspect being the appropriate selection of methodology to behaviour type.

The biologically inspired memory methodology that I have proposed offers a scalable and efficient method for representing knowledge within Artificial Agents that seek to simulate

human behaviour. Efficiency is maintained in a similar method as to that found in biological systems; the occurrence of "forgetting". While it may appear to be a disadvantage for humans to forget, and while certainly not suitable for Artificial Agents not attempting to simulate behaviour, it can be strongly argued that forgetting information allows humans to maintain a relatively small "working set" of memory, and by recreating this phenomenon in software we are effectively creating a method of "garbage collection"; something that has been practised in software engineering for many years.

The memory methodology proposed has enabled the Agent to adjust its behaviour while online. By simulating the process of forgetting, a single behaviour can now be used for handling interaction with a specific object where previously, two behaviours would have been required in order to produce a realistic effect; a single behaviour for finding an object, and another behaviour explicitly stating when and how to forget the object.

Overall, it is the combination of multiple behaviour systems with a Belief network that offers realistic world representation and recall failures that will offer the most compelling and realistic behaviour from an Artificial Agent. By using the most appropriate aspects from each respective system, and overriding the aspects which are flawed with other systems, we are able to avoid some of the common pitfalls that occur in current serious game behaviour simulation.

# CHAPTER 7. FURTHER WORK

## 7. FURTHER WORK

Following on from the work in this thesis, several additional avenues of research could be pursued, with the most immediate gains likely to be found in the biological memory system.

### *7.1 MEMORY LINK SHORTENING*

As memory chunks are used more frequently, it would be beneficial for the chunks to be rearranged in memory so that the number of links from the memory root to the more frequently used chunks is reduced. By performing this reduction, further searches for memory chunks should be faster, as it would not take so many steps to arrive at the most frequently used chunks.

### *7.2 FUZZY LOGIC BASED PERCEPTION FILTERING*

In order to further improve upon giving artificial Agents an "imperfect" view of the world, further work could be undertaken in the fuzzification of the Agent's perception of the world. Objects seen over a distance should be less reliably perceived than nearby objects, sounds can be misheard, and other similar perception flaws that humans suffer from. This would allow the generation of behaviours from the "imperfection" in perception; an Agent could investigate the source of a sound that it is not sure about, or try to move closer to an Object in the distance in order to perceive it more reliably.

### *7.3 BELIEF TRANSFER BETWEEN AGENTS*

Just as humans are able to communicate and share knowledge, Agents should be able to perform similarly. Knowledge from one Agent should be transferrable to another, with this transfer also being affected by some form of filtering to prevent "perfect" transfer.

### *7.4 DYNAMIC CONSEQUENCE GENERATION*

Agents should be able to identify actions within the virtual world that have an identical consequence to Plans that is already knows. By making this link, new Plans could be constructed while the Agent is online, providing multiple Plans to resolve a Desire.

# CHAPTER 8. REFERENCES

Alvarez, G.A., Cavanagh, P., 2004. The capacity of visual short-term memory is set both by visual information load and by number of objects. Psychological Science 15, 106.

Anderson, J.R., Schooler, L.J., 1991. Reflections of the environment in memory. Psychological Science 396–408.

Angelov, P., 2002. Evolving Rule-based Models: A Tool for Design of Flexible Adaptive Systems. Lancaster E-Prints [http://eprints.lancs.ac.uk/perl/oai2] (United Kingdom).

Atkinson, R.C., Shiffrin, R.M., 1976. Human memory: A proposed system and its control processes. Readings in Human Memory 25.

Baddeley, A., 1992. Working memory. Science 255, 556.

Baddeley, A.D., Thomson, N., Buchanan, M., 1975. Word length and the structure of short-term memory. Journal of Verbal Learning and Verbal Behavior 14, 575–589.

Bartish, A., Thevathayan, C., 2002. BDI Agents for Game Development, in: Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems: Part 2. ACM, pp. 668–669.

Baumgarten, R., Colton, S., Morris, M., 2009. Combining AI methods for learning bots in a real-time strategy game. International Journal of Computer Games Technology 2009, 10.

Bhargava, H.K., Branley, W.C., 1995. Simulating belief systems of autonomous agents. Decision Support Systems 14, 329–348.

Bjork, R.A., Bjork, E.L., 1992. A new theory of disuse and an old theory of stimulus fluctuation. From Learning Processes to Connectionist Theory: Essays in Honor of William K. Estes 35.

Bohland, J.W., Minai, A.A., 2001. Efficient associative memory using small-world architecture. Neurocomputing 38, 489–496.

Bratman, M.E., 1987. Intentions, Plans, and Practical Reason. Harvard University Press.

Bratman, M.E., Israel, D., Pollack, M.E., 1988. Plans and resource-bounded practical reasoning. Computational intelligence 4, 349–355.

Braubach, L., Pokahr, A., Lamersdorf, W., 2005. Jadex: A BDI Reasoning Engine. Springer US.

Brom, C., Peskova, K., Lukavsky, J., 2007. What does your actor remember? towards characters with a full episodic memory. LECTURE NOTES IN COMPUTER SCIENCE 4871, 89.

Brown, J., 1958. Some tests of the decay theory of immediate memory. The Quarterly Journal of Experimental Psychology 10, 12–21.

Caises, Y., Leyva, E., González, A., Peréz, R., 2010. An extension of the genetic iterative approach for learning rule subsets, in: Genetic and Evolutionary Fuzzy Systems (GEFS), 2010 4th International Workshop On. pp. 63–67.

Cowan, N., 2001. The magical number 4 in short-term memory: A reconsideration of mental storage capacity. Behavioral and brain sciences 24, 87–114.

Curiel, J.M., Radvansky, G.A., 2002. Mental maps in memory retrieval and comprehension. Memory 10, 113–126.

Delvenne, J.F., Bruyer, R., 2004. Does visual short-term memory store bound features? Visual cognition 11, 1–27.

Dennis, L.A., Farwer, B., Bordini, R.H., Fisher, M., Wooldridge, M., 2008. A Common Semantic Basis for BDI Languages. Programming Multi-Agent Systems 124–139.

Dumont, B., Hill, D.R.., 2001. Multi-agent simulation of group foraging in sheep: effects of spatial memory, conspecific attraction and plot size. Ecological modelling 141, 201–215.

Ebbinghaus, H., Ruger, H.A., Bussenius, C.E., 1913. Memory: A contribution to experimental psychology. Teachers college, Columbia university.

Elizondo, D.A., Góngora, M.A., 2005. Current trends on knowledge extraction and neural networks. Artificial Neural Networks: Formal Models and Their Applications-ICANN 2005 485–490.

Estes, W.K., 1997. Processes of memory loss, recovery, and distortion. Psychological Review 104, 148–169.

Fisher, M., 1994. A survey of concurrent MetateM — The language and its applications. Temporal Logic 480–505.

Genesereth, M.R., Nilsson, N.J., 1987. Logical foundations of artificial intelligence. Morgan Kaufman Publishers, Inc.

Georgeff, M.P., Lansky, A.L., 1987. Reactive Reasoning and Planning. AAAI.

Ghasem-Aghaee, N., Ören, T.I., 2007. Cognitive complexity and dynamic personality in agent simulation. Computers in human behavior 23, 2983–2997.

Gongora, M., Irvine, D., 2010a. ReAd: reactive-adaptive methodology to enable evolving intelligent agents for virtual environments. Evolving Systems 1, 111–127.

Gongora, M., Irvine, D., 2010b. Adaptive intelligent agents based on efficient behaviour differentiation models, in: ANDESCON, 2010 IEEE. pp. 1–6.

Hellmann, M., 2001. Fuzzy logic introduction. Université de Rennes.

Hindriks, K.V., De, F.S., der, W.V., Meyer, J.-J.C., 1999. Agent Programming in 3APL. Autonomous Agents and Multi-Agent Systems 2, 357–401.

Hirvonen, H., Ranta, E., Rita, H., Peuhkuri, N., 1999. Significance of memory properties in prey choice decisions. Ecological Modelling 115, 177 – 189.

Howard, P.J., 2000. The Owner's Manual for the Brain. Bard Press.

Huber, M.J., 1999. JAM: A BDI-theoretic mobile agent architecture, in: Proceedings of the Third Annual Conference on Autonomous Agents. pp. 236–243.

Hugill, A., 2012. Virtual Romans. http://www.ioct.dmu.ac.uk/research/current/virtualromans.html.

Irvine, D., Gongora, M., 2010a. Storage, degradation and recall of agent memory in Serious Games and Simulations, in: Intelligent Systems (IS), 2010 5th IEEE International Conference. pp. 85–90.

Irvine, D., Gongora, M., 2010b. CogBox-Combined Artificial Intelligence Methodologies to Achieve a Semi-realistic Agent in Serious Games. Artifical Intelligence and Soft Computing 492–499.

Irvine, D., Gongora, M., 2011a. Intelligent agents based on a hybrid adaptable structure and a biologically inspired memory, in: Hybrid Intelligent Models And Applications (HIMA), 2011 IEEE Workshop On. pp. 84–89.

Irvine, D., Gongora, M., 2011b. Storing objects in a short-term biologically inspired memory system for artificial agents, in: Computational Intelligence, Cognitive Algorithms, Mind, and Brain (CCMB), 2011 IEEE Symposium On. pp. 1–6.

Khan, I.M., Frayman, Y., Nahavandi, S., 2012. Knowledge extraction from a mixed transfer function artificial neural network. Journal of advanced computational intelligence and intelligent informatics 10, 295–301.

Kuffner, J.J., Latombe, J.C., 1999. Fast synthetic vision, memory, and learning models for virtual humans, in: Proc. CA. pp. 118–127.

Lee, D., Chun, M.M., 2001. What are the units of visual short-term memory, objects or spatial locations? Perception & Psychophysics 63, 253.

Li, Y., Musilek, P., Wyard-Scott, L., 2004. Fuzzy logic in agent-based game design.

Lim, M., 2012. Memory Models for Intelligent Social Companions. Human-Computer Interaction: The Agency Perspective 241–262.

Luck, S.J., Vogel, E.K., 1997. The capacity of visual working memory for features and conjunctions. Nature 390, 279–280.

Magnussen, S., Greenlee, M.W., Thomas, J.P., 1996. Parallel Processing in Visual Short-Term Memory* 1. Journal of Experimental Psychology: Human Perception and Performance 22, 202–212.

Miller, G.A., 1956. The Magical Number Seven, Plus or Minus Two: Some limits on our capacity for processing information. PSYCHOLOGICAL REVIEW 63, 81–97.

Morelli, L.G., Abramson, G., Kuperman, M.N., 2004. Associative memory on a small-world neural network. The European Physical Journal B-Condensed Matter and Complex Systems 38, 495–500.

Murdock, B.B., 1982. A theory for the storage and retrieval of item and associative information. Psychological Review 89, 316–338.

Nareyek, A., 2001. Review: Intelligent agents for computer games. Computers and Games 414–422.

Nuxoll, A.M., 2007. Enhancing intelligent agents with episodic memory. Citeseer.

O'Regan, J.K., 1992. Solving the" real" mysteries of visual perception: The world as an outside memory. Canadian Journal of Psychology 46, 461–461.

Osborne, D., Dickinson, P., 2010. Improving games AI performance using grouped hierarchical level of detail [WWW Document]. URL http://eprints.lincoln.ac.uk/2237/ (accessed 9.11.12).

Page, M.P., Norris, D., 1998. The primacy model: a new model of immediate serial recall. Psychological review 105, 761.

Palmer, J., 1990. Attentional limits on the perception and memory of visual information. Journal of Experimental Psychology: Human Perception and Performance 16, 332–350.

Parunak, H.V.., 1997. " Go to the ant": Engineering principles from natural multi-agent systems. Annals of Operations Research 75, 69–102.

Patel, P., Hexmoor, H., 2009. Designing BOTs with BDI agents.

Peters, C., O'Sullivan, C., 2002. Synthetic vision and memory for autonomous virtual humans, in: Computer Graphics Forum. pp. 743–752.

Phillips, W.A., 1974. On the distinction between sensory storage and short-term visual memory. Perception and Psychophysics 16, 283–290.

Radvansky, G.A., Copeland, D.E., 2006. Walking through doorways causes forgetting: Situation models and experienced space. Memory & cognition 34, 1150–1156.

Radvansky, G.A., Krawietz, S.A., Tamplin, A.K., 2011. Walking through doorways causes forgetting: Further explorations. The Quarterly Journal of Experimental Psychology 64, 1632–1645.

Rao, A.S., 1996. AgentSpeak(L): BDI agents speak out in a logical computable language. Agents Breaking Away 42–55.

Rao, A.S., Georgeff, P.M., 1995. BDI Agents: From Theory to Practice, in: Proceedings of the First International Conference on Multi-agent Systems (ICMAS-95). San Francisco, CA, pp. 312–319.

Rensink, R.A., 2001. Change blindness: Implications for the nature of visual attention. Vision and attention 169–188.

Rensink, R.A., O'Regan, J.K., Clark, J.J., 1997. To see or not to see: The need for attention to perceive changes in scenes. Psychological Science 8, 368.

Rhalibi, A.E.., Merabti, M., 2006. A hybrid fuzzy ANN system for agent adaptation in a first person shooter, in: Proceedings of the 2006 International Conference on Game Research and Development. pp. 54–61.

Rinck, M., Bower, G.H., 1995. Anaphora resolution and the focus of attention in situation models. Journal of Memory and Language 34, 110–131.

Roth, T., Silva, J.A.C. e, Chase, M.H., 2001. Sleep and cognitive (memory) function: research and clinical perspectives. Sleep Medicine 2, 379 – 387.

Saffiotti, A., 1997. The uses of fuzzy logic in autonomous robot navigation. Soft Computing-A Fusion of Foundations, Methodologies and Applications 1, 180–197.

Sardina, S., Silva, L. de, Padgham, L., 2006. Hierarchical Planning in BDI Agent Programming Languages.

Setiono, R., Baesens, B., Martens, D., 2012. Rule Extraction from Neural Networks and Support Vector Machines for Credit Scoring. Data Mining: Foundations and Intelligent Paradigms 299–320.

Siciliano, B., Khatib, O., 2008. Springer handbook of robotics. Springer-Verlag New York Inc.

Silva, L. de, Dekker, A., Harland, J., 2007. Planning with Time Limits in BDI Agent Progamming Languages.

Skubic, M., Noelle, D., Wilkes, M., Kawamura, K., Keller, J., 2004. A biologically inspired adaptive working memory for robots, in: AAAI Fall Symp., Workshop on the Intersection of Cognitive Science and Robotics: From Interfaces to Intelligence.

Sperling, G., 1960. The information available in brief visual presentations.

Spronck, P., Ponsen, M., Sprinkhuizen-Kuyper, I., Postma, E., 2006. Adaptive game AI with dynamic scripting. Machine Learning 63, 217–248.

Spronck, P., Sprinkhuizen-Kuyper, I., Postma, E., 2003. Improving opponent intelligence through offline evolutionary learning. International Journal of Intelligent Games and Simulation 2, 20–27.

Tulving, E., 1972. Episodic and Semantic Memory. Organization of memory 381–402.

Tulving, E., 1985. How many memory systems are there. American Psychologist 40, 385–398.

Vogel, E.K., Woodman, G.F., Luck, S.J., 2001. Storage of features, conjunctions, and objects in visual working memory. Journal of Experimental Psychology 27, 92–114.

Waugh, N.C., Norman, D.A., 1965. Primary memory. Psychological review 72, 89–104.

Wheeler, M.E., Treisman, A.M., 2002. Binding in Short-Term Visual Memory* 1. Journal of Experimental Psychology: General 131, 48–64.

Wickelgren, W.A., 1970. Multitrace strength theory. Models of human memory 65–102.

Winikoff, M., Padgham, L., Harland, J., Thangarajah, J., 2002. Declarative & Procedural Goals in Intelligent Agent Systems.

Wood, O.E., 2004. Autonomous Characters in Virtual Environments: The technologies involved in artificial life in computer games and their affects of perceived intelligence and playability.

Yue, B., Byl, P. de, 2006. The state of the art in game AI standardisation. Murdoch University.

Zadeh, L., 1984. Making computers think like people. IEEE spectrum 21, 26–32.

Zadeh, L.A., 1965. Fuzzy sets*. Information and control 8, 338–353.