Believable Behaviour of Background Characters in Open World Games

Author:
John Fallon

Supervisor:
Dr. Mads Haahr

A dissertation presented to the University of Dublin, Trinity College in fulfilment of the requirements for the Degree of

Master of Science in Computer Science
(Interactive Entertainment Technology)

University of Dublin, Trinity College

August 2013

Declaration of Authorship

I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this, or any other University, and that unless otherwise stated, is my own work.

Signed:

Date:

Permission to Lend and/or Copy

I, the undersigned, agree that Trinity College Library may lend or copy this dissertation upon request.

Signed: _____

Date: _____

Acknowledgements

First and foremost, I would like to thank my supervisor Dr. Mads Haahr for his advice and guidance in the completion of this project.

I would like to give a special thanks to Adam Russell whom I had the pleasure of meeting and whose insight proved invaluable in understanding the problem domain.

Thanks to all the lecturers and demonstrators on the course for their time and commitment.

Finally I would like to thank my family and friends for their tremendous encouragement and support.

Abstract

This dissertation explores the notion of believability of non-player characters (NPCs) in "open world" games. The focus lies specifically on "background" characters which exist simply to provide a backdrop for the game. It is hoped that with increased believability, the video game industry could deliver even more engaging and immersive gameplay experiences. A hypothesis was envisaged which suggested that a model to drive the character's behaviour may enhance its overall believability. The proposed model considers two dimensions of behaviour, mental-physical state and personality. This is realised through the use of a State Vector, an instantaneous evaluation of the NPCs mental and physical states, and a collection of State Modification Vectors, that capture how particular activities affect the mental and physical state of the NPC. At the core of the model is the Activity Selection Manager which drives NPC behaviour. In order to test the hypothesis a prototype of the model was implemented in The Elder Scrolls V: Skyrim [1] with the editor used to create content for the game, known as the Creation Kit. The implementation showcases a promising model with much potential where characters gain a unique sense of personality and become much more interesting to encounter in the game. However, it remains to be seen if the model truly offers a tangible improvement over existing approaches. Future work is necessary to procure more conclusive evidence.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This dissertation explores the notion of believability of non-player characters (NPCs) in "open world" games. The focus lies specifically on "background" characters which exist simply to provide a backdrop for the game. The research is especially concerned with the existing approaches to character behaviour and the effect they have on the observed believability of the character. A hypothesis was envisaged which suggested that a model to drive the behaviour of the character may enhance its overall believability. In order to test this hypothesis a prototype of the model would be implemented in an open world game where the resulting character behaviour would be observed and compared to that already in the game. It is hoped that with increased believability, the video game industry could deliver even more engaging and immersive gameplay experiences.

1.1 Motivation

The development of many of today's videogames is primarily focused on the audiovisual aspect of gameplay leaving other areas overshadowed [2–7]. Some researchers claim that Artificial Intelligence could offer something more. Castronova states that "of all the technological frontiers in world-building, artificial intelligence (AI) holds the most promise of change" [8]. Bartle expresses the potential of NPCs in games stating that "from the point of view of world design, AI promises great things. If virtual worlds could be populated by intelligent NPCs, all manner of doors would open" [9]. Similar views have also been expressed by leading industry figureheads such as Warren Spector [10]. These opinions highlight the vital role

of NPCs in making the virtual world seem alive. The feeling of immersion, defined by Bartle as "the sense that a player has of being in a virtual world", is key to establishing the sense of an inhabited world [9]. This feeling is related to the concept of *willing suspension of disbelief* [11], i.e. the player's decision to be fully engaged in the game regardless of some noteworthy details that lack realism. In order to assist this process and maintain suspension of disbelief, it is imperative that NPCs behave such that they appear to be part of a living world. If NPCs act out of character, the realism of the game is shattered and player immersion can be lost. Therefore, it is required that NPCs have believable behaviours.

In essence, believability is the subject of phenomenology, the study of structures of consciousness as experienced from a first-person viewpoint [12]. The concept of believability has for quite some time been studied and explored in various domains including literature, theatre, film and radio drama amongst others. In fact, traditional character animators such as the Disney animators of the 1930s are among those who have strived to create believable characters and who have made significant progress in achieving this goal. Character believability tends to have steep requirements [13] including features such as personality, emotion, self-motivation, social relationships, consistency, the ability to change and the ability to maintain an "illusion of life", by possessing goals, reacting and responding to external stimuli, etc [13]. The term "believable agents" has been used to describe believable *interactive* characters and has seen continued interest ever since Bates and Reilly promoted this concept in their 'Oz project' [14, 15]. It is now generally agreed that personality and the expression of emotion are key factors of believable agents.

In videogames, the complexity of NPC AI differs amongst the various game genres. A character that is central to the story of a game, such as Alyx Vance in *Half Life 2* [16], requires a much more sophisticated level of AI than that of a simple "background" NPC in a game such as *The Elder Scrolls V: Skyrim* [1]. Therefore, it could be said that believability in games is intrinsically linked to the purpose of the characters in the game.

Figure 1.1 illustrates various stages of NPC AI complexity. On the lower end of the scale is Fargoth, a notably encountered NPC in Morrowind [17] whose main purpose in the game is to provide the player with a couple of simple quests. Fargoth displays little interaction with the game environment and other than having a few topics of conversation, has limited interaction with the player. On the other end of the scale is Elizabeth, an AI companion character in Bioshock Infinite [18].

Elizabeth is a much more believable and lifelike character who is fully voiced, reacts to the environment around her and to the player's actions. When talking to the player or commenting on the world, she does so with clear facial expressions that reflect her emotional state. Elizabeth is not just an observer of the game world but also a reliable partner in the experience. Issuing weapons and supplies such as health and ammo and never getting in the way, Elizabeth makes the player feel as though she is there for them, not either a burden or providing an unfair gameplay advantage [19]. Elizabeth is a prime example of NPC AI at the highest level of complexity.

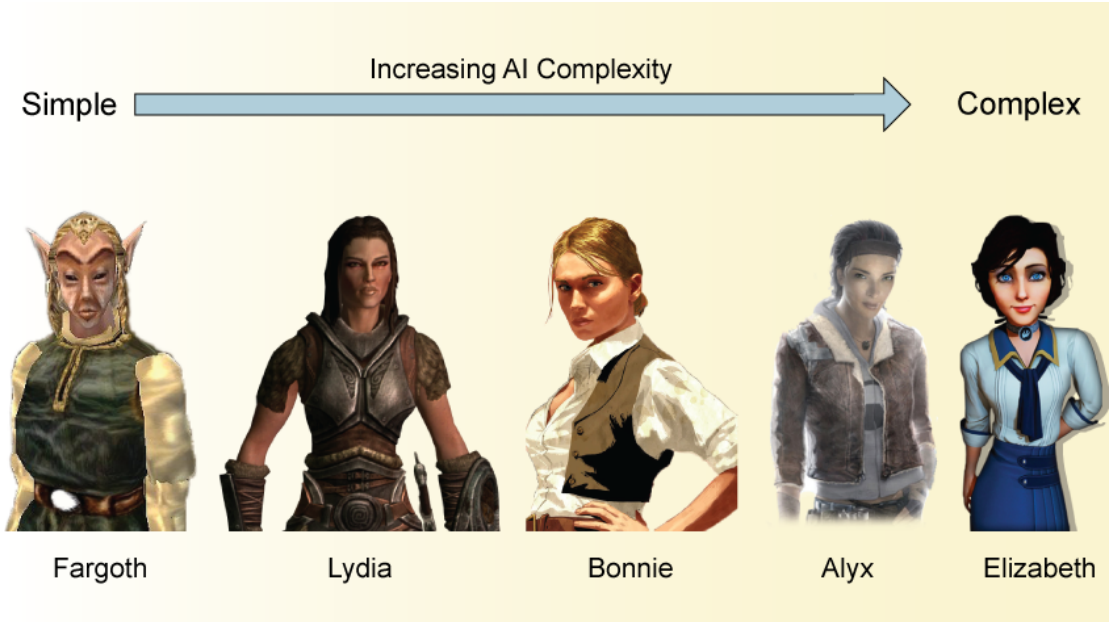


FIGURE 1.1: NPC AI Complexity Scale. From left to right: Fargoth, (Morrowind [17]) adapted from [20]. Lydia (Skyrim [1]), adapted from [21]. Bonnie MacFarlane (Red Dead Redemption [22]) adapted from [23]. Alyx Vance (Half Life 2 [16]) [24]. Elizabeth (Bioshock Infinite [18]) [25]

The "open world" style of gameplay, which has become very popular [26, 27], is one in which the role of NPCs is quite prominent and thus player expectation of NPC behaviour is quite high, i.e. the NPCs should be less static and more human-like. This is a problem that has proven extremely difficult to solve, or as Adams states: "Simulating human beings is the most difficult and also the most important problem in game design" [28]. Open world games require a large number of NPCs in order to populate the virtual environment and to create the illusion of an inhabited world. Given the large quantity of NPCs that are required to populate these expansive virtual worlds, it is too time consuming for developers to hand-craft every single NPC with completely unique behaviours. As a consequence,

behaviour of background NPCs in open world games is often repetitive, predictable and thus not very believable.

1.2 Improving Believability

Typical industry methods for creating believable interactive characters rely heavily on scripting, where hand-coded animation procedures, hard-coded behaviours, voice acting and dialogue scripts are used to personify the intended character. Scripting like this, however, is arduous and rigid causing NPCs to look very static and which often only interact with the player through certain actions or when a dialogue process is initiated by the player. An alternative, though not as simple as it may sound, is to use AI algorithms and graphics techniques to enable character behaviours to adapt to the outcomes of interaction. Researchers have worked on several fronts in order to create believable expressive characters that are capable of dynamically adapting within interactive narratives. For example, graphics researchers have attempted to incorporate emotion and personality as parameters to alter the animations of a virtual character, while artificial intelligence researchers have focused on integrating models of emotion and personality to create characters that appear to have the ability to improvise [29].

1.3 Objectives

The purpose of this project is to design, implement and evaluate a new model to support believable behaviour of background characters in the context of open world games. To achieve this, a number of objectives that must be met are defined, whereby the model must:

(a) Offer a tangible improvement over existing approaches.

(b) Support procedural content generation in order to reduce developer workload.

(c) Lend itself to efficient implementation in order to accommodate CPU and memory usage.

The focus of the project is on objective (a) due to its fundamental significance, that is, to create better or more realistic NPC behaviour than that which already exists within the game. This is of utmost importance as it tests the effectiveness of the model. Objective (b) refers to the automatic or random generation of concepts that would otherwise have to be managed by the developer. This takes precedence over objective (c), which states that the model should be efficiently implementable, since the project is a proof-of-concept study concerned with implementing just a prototype of the model.

1.4 Dissertation Roadmap

The forthcoming Chapter 2 reviews the state of the art in a number of areas, including believability, NPC models and AI architectures for NPC behaviour. Chapter 3 details the design of the proposed model to address believable behaviour of human-like NPCs in open world games. Following this, in Chapter 4, is a discussion of the implementation of the project. Chapter 5 presents an evaluation of the successes and failures of the project. Finally, in Chapter 6 a conclusion is drawn, detailing the contributions made by the project and a proposal is established for any future work that could be undertaken with this project.

Chapter 2

State of the Art

The purpose of this chapter is to complete a review of the state of the art in a number of areas that are of significance to this project. Starting with the topic of "believability", a brief definition of the term is presented, followed by a discussion that aims to provide an understanding of human behaviour. Also falling into the category of believability is the subject of "believable agents". To assist in the conception of a new model for believable behaviour of NPCs, an analysis of the requirements for believable agents is undertaken. The next section investigates the current state of NPC models taking into account the theories about human behaviour, providing an insight into how NPCs can be made to act more "human-like". Finally, the last section discusses AI architectures for NPC behaviour which are required to support the straightforward creation and reuse of sophisticated character behaviours.

2.1 Believability

Believability is fundamentally a phenomenological construct. Literally, phenomenology is the study of "phenomena": appearances of things, or how things appear in our experience, or the ways in which things are experienced, thus the meanings things have in our experience. However phenomenology can simply be defined as the study of structures of experience or consciousness. This includes a range of types of experience including action, desire, emotion, imagination, perception, thought and volition [12].

2.1.1 Understanding Human Behaviour

Maurice Merleau-Ponty was a French phenomenological philosopher who conceptualised a model that captures the human capacity for action. Merleau-Ponty's view is considered by many to be better than any other account and is also claimed as one that should be adopted by anyone looking to gain an understanding of human action. Merleau-Ponty's account of unreflective behaviour is of particular interest. What is considered as unreflective behaviour depends on one's interpretation of what it is to reflect. In this context, to designate a particular behaviour as 'unreflective' means to characterise a specific part of one's participation in it. In this regard, behaviour is unreflective if it is experienced by the subject without the assistance of thought. To illustrate the phenomenon, take for example the case of someone inattentively wandering into the kitchen to make a cup of tea whilst contemplating about what to eat for lunch, this would be considered as behaving unreflectively. Of course, it is well-documented that conscious thought about what one is currently doing inhibits the capacity to do it; thinking hinders the 'flow' of the action. Merleau-Ponty maintains the view that unreflective behaviour is caused by the agent's perception's of its environment [30].

Considering Merleau-Ponty's view, the objects of perception are things that are of particular interest to the perceiver in terms of his/her ability to interact with them. Regarding perception, one's surroundings are instantly presented as 'needing' or 'implying' a particular type of behaviour such that the perceiver does not confront things that simply have objective qualities such as shape, size, etc., but rather with objects that are, for example, throwable, kickable, edible, etc. The behaviour that is perceived to be required by one's environment will be behaviour that correlates with one's current task [30]. Consider the following example:

For the player in action the football field is not an 'object'... It is pervaded by lines of force (the 'yard' lines; those which demarcate the 'penalty area') and articulated into sectors (for example the 'openings' between the adversaries) which call for a certain mode of action... [31]

Here, Merleau-Ponty explains that for a player absorbed in a football game, the pitch is recognised as a space, delimited by particular objects, that presents the player with opportunities to execute specific actions. The yard lines and those that designate the penalty box are not perceived to simply exist as white lines in a precise point in space. They are instead perceived as real boundaries that

designate parts of the pitch that are significant for the player's behaviour. The gaps between the rival players are not perceived as simply unoccupied space, but rather as 'openings', or in other words, opportunities or a chance to reach the goal or to pass the ball to another teammate. Furthermore, it is only when engaged in a game of football that the pitch is perceived like this by the agent. If perchance they wandered onto the pitch while out for a walk, the ball will not be seen as to-be-intercepted, but as to-be-avoided [30].

With Merleau-Ponty's view, unreflective behaviour is initiated and controlled by the perceived opportunities for action. An opportunity to behave is simply perceived by the agent, who reacts by behaving so, without requiring any intermediary states representing their involvement in the activity. Therefore, in the previous examples, when one inattentively wanders into the kitchen to make a cup of tea, the kettle is perceived as for-boiling-water, the mug as for-containing-tea, the teabag as for-brewing, etc. These perceptions 'draw forth' from the individual the action of making tea whilst thinking of other things. Dreyfus [32] defines action that is instantly brought about by the agent's perception of its surroundings as 'absorbed coping' [30].

Given the above analysis of absorbed coping, action generated by thought can be explained. In other words, how intentions can cause certain behaviour. As described previously, unreflective behaviour is generated by perceptual experience. The composition of perceptual experience is determined by the agent's current activity, together with its surroundings and the motor-skills it possesses. Essentially the agent perceives its surroundings in consideration of its current task and thus perceives opportunities to perform the skills that are suited to the current activity. Humans can make a conscious decision to carry out a task or activity, and deciding to complete X involves forming the intention to carry out X. One decides to perform the activity of, for example, playing music by forming the intention to do so. Since one's current activity affects the process of absorbed coping and one decides to carry out one's current activity by establishing the intention to do so, it would seem that the earlier account of unreflective behaviour has already explained how intention can create action [30].

The above assumes that one carries out the activity X through forming the intention to do X. This, however, is not always the case. Physical beings have needs to satisfy such as hunger, sleep, avoiding danger etc. These needs dictate tasks that must be carried out by the living creature. If, for example, it feels hungry, this

feeling appoints the activity of acquiring food. An intention to carry out the activity of satisfying hunger is not necessarily formed, but rather the activity is carried out through the very feeling of hunger. A living creature can in fact progress from one activity to the next without even making a decision. At all times it would be involved in a specific activity, and when completed, another activity would be engaged in due to the creature's current state. For example, it has finished the activity of acquiring and eating food, its hunger is satiated but now it feels sleepy, and thus engages in the activity of sleeping. Humans are physical beings that instinctively satisfy hunger, sleep, etc. Therefore, humans could potentially go through life while being simply engaged in the activities that are brought about instinctively. Given that humans carry out activities that are imposed upon them due to the fact that they are physical beings, yet they can also carry out activities through forming intentions to do so, it can be questioned what forms the possibility of a human deciding to accomplish X, instead of simply procuring food when hungry, sleeping when tired, etc [30].

2.1.2 Believable Agents

In order to devise a generic model for the believable behaviour of NPCs, the requirements must first be identified and then built into the agent. Loyall [13] has defined a widely accepted set of criteria for believable agents through an analysis that draws from two main sources, character-based artists and experience with agents.

- Personality: Regarded as the single most important requirement for believable agents, personality is defined as *"all of the particular details - especially details of behaviour, thought and emotion - that together define the individual"* [13]. Essentially, it is the details that work together to bring the character to life. Loyall [13] explains that an abundance of these carefully selected individual details can make for very powerful characters. Take for example two characters in a movie who both drink but for apparently different reasons and in dramatically different ways, walk or talk differently, have different desires or interact differently with other characters. Now imagine how less powerful, less believable, the characters would be if their behaviours were alike. Loyall [13] claims that artists who create characters also share

this view, as do writers and playwrights, who need to have a deep understanding of their characters to ensure the characters' personality is portrayed through their actions, dialogue and responses. Seemingly the lack of tridimensional characters is a major flaw in the creation of good characters in drama. "Tridimensional characters" refers to those that are created by specifying details along three dimensions: physiology, sociology, and psychology. These must all be known in order to know what the character will do in a certain scenario, but will be portrayed through the behaviour of the character. In order to create believable agents that appear as real as these characters, a personality that exhibits the individuality as presented by these traditional characters must be given to the autonomous agent.

- Emotion: Just as in the traditional character-based arts, as well as acting, where emotion is regarded as a fundamental requirement to bring the characters to life, so too is it in making believable agents. In order for an agent to be believable, it must appear to have emotions and be capable of expressing these emotions in a manner that corresponds with its personality.

- Self Motivation: In order to create believable agents, a requirement is that they should be self motivated. In other words agents should be able to carry out actions of their own accord and not just react to stimuli, as is often the case in the creation of autonomous agents. The power of this requirement is quite substantial and can lead to increased admiration and interest in the agent when it appears to complete tasks according to its own intentions.

- Change: Essential to the believability of agents is how they grow and change. However, this growth and change cannot be arbitrary, rather it should correspond with the agents personality. For example, as a result of interaction, a character may grow to like or dislike the human interactor. The growth itself may be automatic but the liking or disliking would reflect the personality of the character.

- Social Relationships: Since humans lead a generally social lifestyle, an important requirement for believable agents is social relationships. There should be interaction amongst agents, influenced by the relationships that exist between them, and in turn those interactions may influence the relationships.

- Consistency of Expression: A basic requirement for believability is the consistency of expression. An agent can have a multitude of behavioural and

emotional expressions such as facial expression, movement, body posture, voice intonation etc. In order to be believable at all times, these expressions must work in unison to communicate the relevant message for the personality, feelings, thinking, situation etc. of the character. If this consistency is broken, the suspension of disbelief is likely to be lost.

- The Illusion of Life: Some requirements necessary for believable agents are overlooked or taken for granted by the conventional media. For example, it is not necessary to specify that an actor should be able to speak while walking since any human actor reading the script can do this easily. This is simply not the case when creating believable agents where those properties must be built into the agent. A list of such properties, which altogether form this final requirement for believability, includes: appearance of goals, concurrent pursuit of goals and parallel action, reactive and responsive, situated, resource bounded, existence in a social context, broadly capable and well integrated (capabilities and behaviours).

2.2 Current NPC Models

Given that NPCs have a very limited set of behaviours, it is important to take note of the theories about human behaviour. This provides an insight into how NPCs can be made to act more "humanlike" as well as an understanding of the fundamental concepts involved. In psychology (and other disciplines), various competing theories exist on human behaviour including theories on human necessities, motivational processes, social comparison theory, social learning theory, theory of reasoned actions, etc. Many of these theories take into account a portion of all possible human behaviour and the settings where this behaviour occurs [33].

2.2.1 Practical Reasoning

In most agent systems, the internal model of the agent is built on a Beliefs-Desires-Intentions (BDI) architecture [34]. This model takes root from the theory of practical reasoning, comprising of two main procedures: deliberation (deciding what states of affairs should be accomplished) and means-ends reasoning (determining how to accomplish these states of affairs). A separation between plan selection

and execution is provided by the architecture enabling BDI agents to evenly spend time deliberating about plans and executing those plans. A BDI agent determines intentions from its beliefs and desires and chooses a group of actions to accomplish these intentions. Therefore, desires are the primary source of behaviour. However, this does not provide any guidelines for designing agents with believable behaviour. The dilemma of the BDI model is that it only accounts for practical reasoning through the connection between deliberation and means-ends reasoning and does not offer any theoretical rationale for answering questions such as: What should an agents desires be (and why)? Why should a specific desire be chosen over another? In addition, BDI agents generally do not learn from experience or adapt to their interaction environment [35].

2.2.2 Decision Making

A meta-model can be used to combine some of the psychological and social psychological theories into one framework. One such meta-model is the Consumat approach [33].

With the introduction of the Consumat [36] [37], a generic model of human behaviour was formed based on the decisions that are made by people to satisfy their basic necessities in several settings. Based on the literature on consumer behaviour, this model provides a simulation framework that captures some of the fundamental behavioural principles. The model allows for the simulation of a number of important procedures that all together capture human decision making in various scenarios including consumers purchasing products, farmers choosing a crop, people deciding on a place to live, and other scenarios where people choose a behaviour from a range of possibilities. However, this model lacks the capability to simulate intricate cognitive processes, logical reasoning and morality in agents [38].

As mentioned previously, the Consumat model gracefully combines many of the dominant psychological theories on human behaviour. They are categorized into a 2*2 matrix. On one hand based on the level of need of satisfaction (LNS) and behavioural control (BC) while on the other hand based on certainty, type of needs and cultural perspective (CP). With regards to the amount of certainty experienced by the agent, it can be either confident in its ability to make decisions (therefore establishing an individual perspective) or unsure (therefore leading to

the consultation of others). If the agent has a high need for behavioural control and a high level of need of satisfaction, the amount of processing required is reduced (using automated reactions), while a need for cognitive processing results from a low level on both. This yields four common approaches humans follow, specifically, repetition, deliberation, imitation and social comparison [33].

	Automated reactions (high LNS, high BC)	**Reasoned (re)actions** (low LNS, low BC)
Individual determined behaviour (certainty, personal needs, private individualist CP)	REPETITION Classical and operant conditioning theory	DELIBERATION Decision and choice theory, theory of reasoned/planned behaviour (attitude and perceived control)
Socially determined behaviour (uncertainty, social needs, egalitarian CP)	IMITATION Social learning theory, theory of normative conduct.	SOCIAL COMPARISON Social comparison theory, relative deprivation theory, theory of reasoned/planned behaviour (social norm)

TABLE 2.1: The Consumat Model. A classification of eight major theories on human behaviour. LNS = level of need satisfaction, BC = Behavioural Control and CP = Cultural Perspective (adapted from [36]).

Utilizing the Consumat model, NPC developers are provided with the means to form a general framework of concepts to develop NPC behaviour and also a solution for the problem on how to switch between various modes of behaviour. However, due to the fact that it is a meta-model based on psychological and social psychological theories, social aspects of behaviour are not addressed in the same way that they are in social sciences. In order to create NPCs that accurately behave like humans, a social theory perspective is required [33].

2.2.3 Social Behaviour

In social sciences, ample theories exist on human behaviour. A meta-model developed by Carley and Newell [39] presents a Fractionation Matrix (C&N matrix) on social behaviour to provide an insight and an understanding of the complexity of agents. The C&N matrix is based on a range of (mostly) sociological theories that

can be used to envisage and classify human-like behaviour in agents. The aim was to create a "Model social agent" (MSA) an agent with strong, human-like social behaviour [33].

Comprising 74 cases of social behaviour including "goal directed" and "crisis response", the matrix (an adapted version can be seen in figure 2.1) illustrates the intricacy of AI behaviour in various social scenarios. The X-axis in the matrix depicts Knowledge, i.e. the knowledge required by an agent for increasingly advanced situations. This ranges from non-social tasks (for example, cutting wood) where there is a relatively low demand on knowledge and the agent behaves on accurate information, to more aesthetic cultural behaviour and perspective (for example, upholding norms) where the agent behaves on imprecise information. Processing is depicted on the Y-axis, where the agent ranges from being omniscient/omnipotent (OA, top left) where information does not need to be collected nor tasks reflected upon, to an emotional cognitive agent (ECA, bottom row) where, in theory, the agent lets "emotions modify and limit the behaviour of the cognitive agent" [39]. Progressing down and to the right in the matrix sees the agent become more human-like [5].

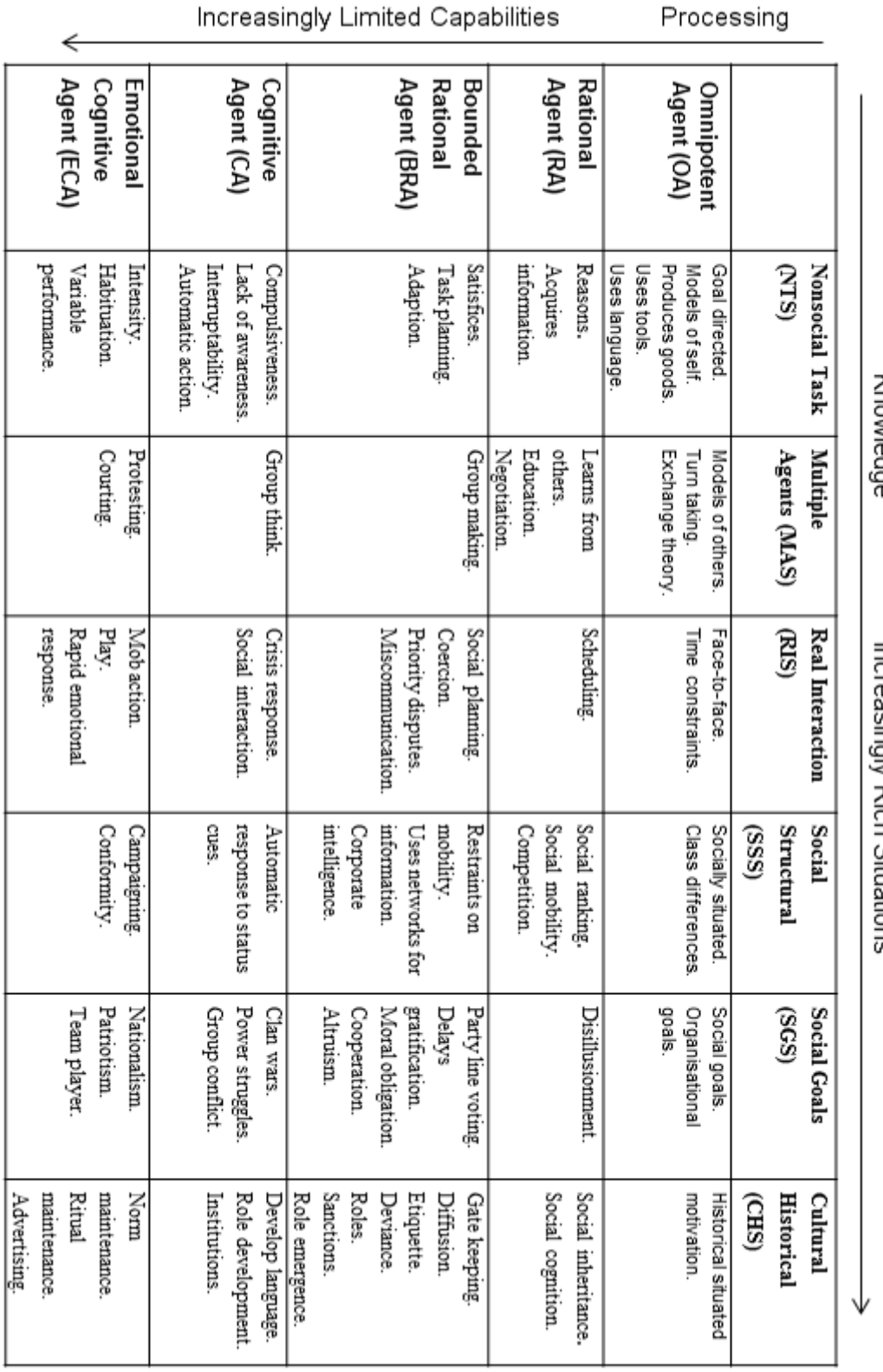
In comparison to a MSA, the majority of NPCs would in fact be limited in both knowledge and processing abilities. Observing the knowledge situation (the X-axis in the matrix), there is no cooperation or communication amongst NPCs. The significance of this situation is that removing other agents results in the non-social task situation of Figure 2.1, which is without social content. This has drastic consequences on the capabilities of NPCs. On the Y-axis NPCs are harder to find. It is important to note that the environment has an impact on the actions of an agent and that more complex goals are enabled through the mental model of the agent. NPCs have characteristics akin to the bounded rational agent. They are rational in their attempts to attain their goals and they have a limited attention, causing difficulty for the agent to process all information in its task environment. However, NPCs are lacking some elements of the bounded rational agent, the cognitive agent and the emotional cognitive agent. NPCs generally lack a memory function and that would make NPCs a class of its own in the matrix. Considering some typical scenarios of NPC interaction, patterns that are not unlike the behaviour of state machines can be identified, where the NPCs typically behave according to various kinds of stimuli. The majority of NPCs could nevertheless be described with traits from either or both the cognitive agent

and the bounded rational agent thus prompting the placement of NPCs in the matrix which also corresponds with the position of the Consumat model and its supporting theories [33].

It is quite apparent that NPCs are deficient of numerous attributes required to exhibit greater complexity. With the addition of a memory function with set learning abilities, the NPCs capacity for interaction would be substantially improved, even for simple tasks they cannot yet perform. To augment the NPCs and therefore shift them from the Non-social task situation in the C&N matrix, it is necessary for agents to have greater knowledge about their task environment as this affects the complexity of an agents goals. To develop a social system of agents, a model is required to distinguish what aspects of interaction should be incorporated into the system. Also, a means of trust and norms between agents is required to enhance the believability in terms of choosing other agents to cooperate with [33].

A study was carried out [2] that combined the matrices of Johansson & Verhagen [33] and Carley and Newell [39], producing a matrix with a total of 80 values, as illustrated in figure 2.1. The study was completed to determine the matrixs viability for assessing NPC social capability in games. Using this work as a basis, additional studies were carried out to further investigate the topic. This included analysing a number of games where NPCs play an important role in maintaining the players enjoyment of the game and then establishing anti-heuristics that could be used to determine non-desirable behaviour in NPCs [5] as well as to devise design patterns that might be useful in identifying similar problems in existing games [6].

In describing the needs for more intelligent NPCs, analysing the current behaviour set and proposing a framework to make this behaviour set more flexible, Johansson & Verhagen [33] formulated an architecture for more believable NPCs referred to as the Model Social Game Agent (MSGA). The MSGA, based in part on Carley and Newells [39] MSA, aims to solve some inadequacies common in NPCs thus enabling more believable NPCs to be developed. Similar to most agents, the MSGA (figure 2.2) requires information in order to make decisions. However, compared to most NPCs, the input from the world is evaluated differently. The MSGA differs from NPCs in that there exists an emotional state and a social state that contribute to a behaviour selection, which subsequently leads to an output from the agent [4].



	Nonsocial Task (NTS)	Multiple Agents (MAS)	Real Interaction (RIS)	Social Structural (SSS)	Social Goals (SGS)	Cultural Historical (CHS)
Omnipotent Agent (OA)	Goal directed. Models of self. Produces goods. Uses tools. Uses language.	Models of others. Turn taking. Exchange theory.	Face-to-face. Time constraints.	Socially situated. Class differences.	Social goals. Organisational goals.	Historical situated motivation.
Rational Agent (RA)	Reasons. Acquires information.	Learns from others. Education. Negotiation.	Scheduling.	Social ranking. Social mobility. Competition.	Disillusionment.	Social inheritance. Social cognition.
Bounded Rational Agent (BRA)	Satisfices. Task planning. Adaption.	Group making.	Social planning. Coercion. Priority disputes. Miscommunication.	Restraints on mobility. Uses networks for information. Corporate intelligence.	Party line voting. Delays gratification. Moral obligation. Cooperation. Altruism.	Gate keeping. Diffusion. Etiquette. Deviance. Roles. Sanctions. Role emergence.
Cognitive Agent (CA)	Compulsiveness. Lack of awareness. Interruptability. Automatic action.	Group think.	Crisis response. Social interaction.	Automatic response to status cues.	Clan wars. Power struggles. Group conflict.	Develop language. Role development. Institutions.
Emotional Cognitive Agent (ECA)	Intensity. Habituation. Variable performance.	Protesting. Courting.	Mob action. Play. Rapid emotional response.	Campaigning. Conformity.	Nationalism. Patriotism. Team player.	Norm maintenance. Ritual maintenance. Advertising.

FIGURE 2.1: Carley and Newell Fractionation Matrix (adapted from [39]).

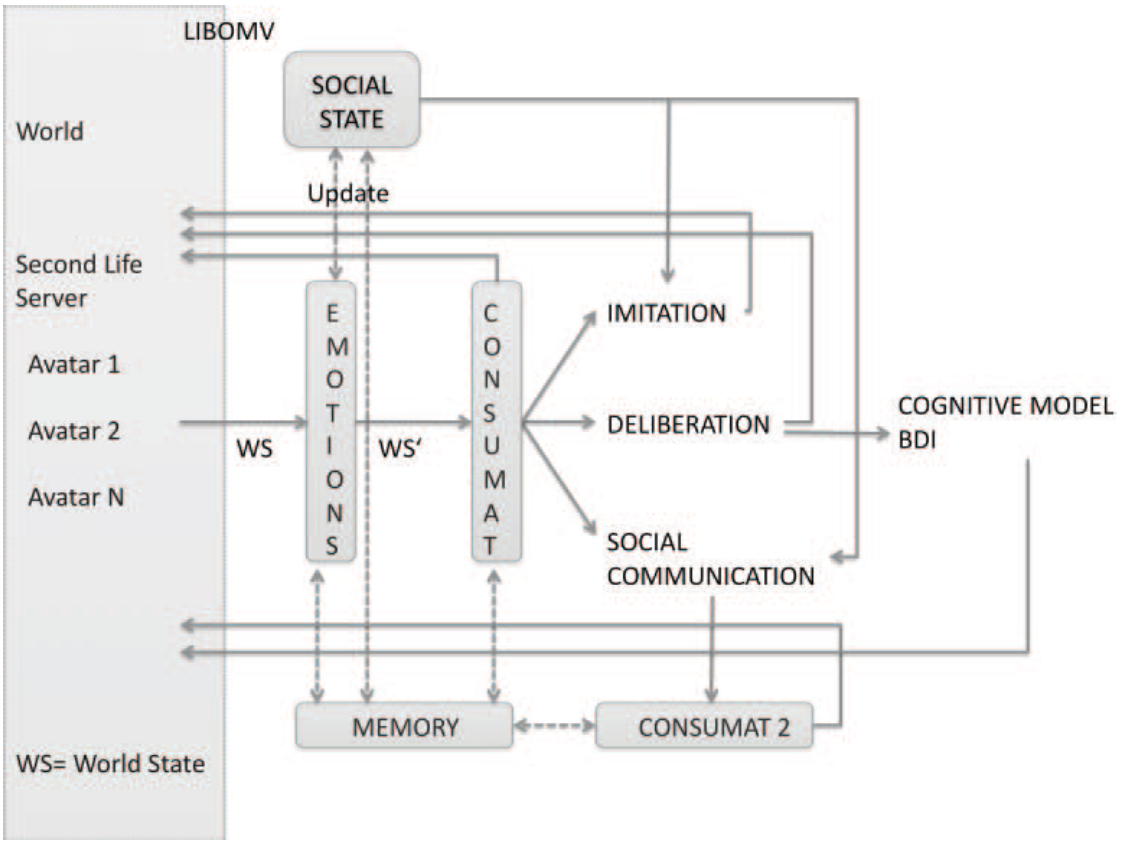


FIGURE 2.2: Model Social Game Agent - MSGA ([4]).

A key component of the MSGA is a memory module that essentially monitors three aspects of the agent: Social state, Emotions and evaluations in the Consumat. Perception of the world occurs through sensory input (WS) which gets passed straight to the emotions module. Prior to passing the processed input (WS') on to the Consumat, the MSGA accesses memory from previous emotions and social state. This is when the Social state of the MSGA is updated. The input from the emotions module is then evaluated by the Consumat which, depending on certainty, determines a strategy. If the information is evaluated with a high level of certainty, the Consumat adopts an automated action and repeating previous actions. However, if the input from the emotions module is evaluated with a low level of certainty, one of three possible strategies will be selected: imitation, deliberation or social communication [4].

2.2.4 Embodiment & Situatedness

More comprehensive frameworks have been recently proposed with a focus on serious games. Inamura et al [40] propose a simulator environment known as SIGVerse

which includes a combination of dynamics, perception and communication simulations, with a background in robotics simulation and therefore with a strong model of embodiment and situatedness. Embodiment involves equipping a body with sensors and actuators which enables structural coupling with the agents surroundings. A video game agent can therefore be said to be embodied in that sense (or 'virtually embodied') since it procures exteroceptive and proprioceptive sensing data through its software sensors and actions can also be performed using its software actuators. Situatedness is concerned with interaction with the world. Take for example a video game character, if it interacts with the simulated world, the game environment is affected, which in turn affects the agent [41]. The SIGVerse project does not however incorporate social interaction very well.

With a background in military simulations, Ball et al [42] implemented The Synthetic Teammate project with the goal of developing a cognitively plausible, yet functional synthetic teammate. At its core is a cognitive architecture instead of the BDI model often used in agent-based models. Included is interaction with the physical environment, language comprehension and generation (via chat) as well as dialogue management for the language-based communication. Additionally there is a situation modelling component which represents the current situation and constitutes the fundamental meaning representation of the system. All components are rooted in psychological theories and models. The problem however lies in the fact that the number of team members is low and the task is quite specific leading to believable interaction in a confined domain. More generic tasks lead to the language parts becoming really complex. In the context of open world games, this is likely to be a vital problem to solve.

Research carried out by Ijaz et al [43] addresses believability in the context of NPC - human interaction. More specifically, it focuses on the integration of believable conversation, including facial expressions, gestures and body movements, believable awareness including awareness of the environment, self and interaction. Additionally, using the concept of virtual institutions, a normative layer is included. However, it is lacking any NPC (or even human) decision making on the social level. The normative layer is regarded as being separate from the agents and is therefore not open for dynamics from the agents. Essentially, it is impossible to break norms [34].

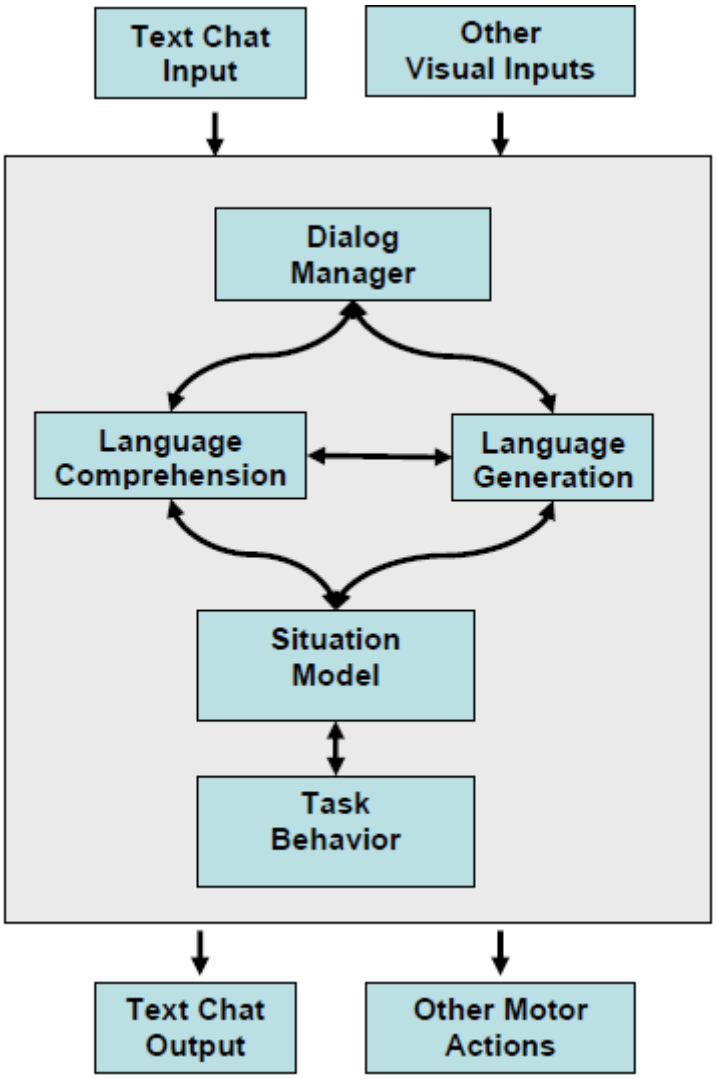


FIGURE 2.3: Synthetic Teammate Overview ([42]).

2.2.5 Morality

A model for moral agents to be used in the context of serious games has been presented by Coelho and da Rocha Costa [44]. The focus is on a moral system, mapping in part to norms (specifically, moral norms) leading to meta control actions over the range of behaviours under consideration of the agent. Essentially, the proposed model operates as a filter after cognition but with emotions parallel to cognition. Any automatic reactions that are not based on emotions are effectively omitted and the role of the judgement part is limited to being a filter of actions, compared to norms in a broader sense which often even produce actions or goals. The model however, lacks a distinct correlation to social theories and social theory meta models [34].

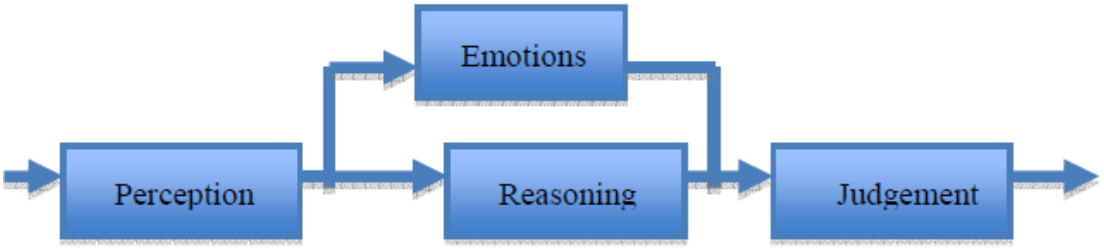


FIGURE 2.4: Simplified sketch of a moral creature ([44]).

2.3 AI Architectures for NPC Behaviour

In order to support the straightforward creation and reuse of sophisticated character behaviours, games require an AI architecture. Cutumisu & Szafron [45] have identified five features that behaviours should exhibit. A behaviour should be *responsive* (promptly react to the environment), *interruptible* (impedible by other behaviours or events), *resumable* (continue after being interrupted), *collaborative* (instigate and react to joint behaviour requests), and *generative* (easily implemented by non-programmers). However, many AI architectures fail to support at least one of the five features [45].

The most common game techniques for administering NPC behaviour include the use of *Finite state machines* (FSMs) [46], *hierarchical state machines* (HSMs) [47] and *behaviour scripts*. Two significant impediments to FSMs include the inability to save state that can be returned to later and their deficiency in identifying high-level patterns that are reproduced regularly such as sequences and conditionals. Thus, FSM techniques do not scale well and their representation means that behaviours are hard to outline. Collaboration is especially difficult to code using FSMs. Various extensions to FSMs exist that support more complicated behaviours, including *Stack-based FSMs* (for interruption and return), *Fuzzy State Machines*, *Hierarchical FSMs* (the use of super-states contribute to sharing and reuse of transitions), and *Probabilistic FSMs*. FSMs and their extension HSMs lack resumability. A recent game that utilises FSMs is *Assassins Creed* [48] (in which the behaviours of the guards are easy to predict). HSMs have a disadvantage in that priorities are not reflected explicitly by the hierarchy and structure of the graph. A lack of support for priorities means that the responsive feature is difficult to attain. [45].

Behaviour Trees [49] (BTs, or hybrid HSMs) and *planners* have been used in some recent games. Behaviour trees support reuse by making individual states modular.

Essentially generalising HSMs, they enable programmers to add code in arbitrary nodes at any level in the HSM tree. The overall priorities are revealed by the tree much more intuitively than FSMs and with a more discernible behaviour hierarchy: behaviour sequences are mid-level while the behaviour selector is at the top level. Joint behaviours are implemented in the *Halo* series [50] using BTs, however, they are not resumable. HSM/BTs are used in *Left 4 Dead* [51], however, the games allies do not follow the player character (PC) despite the fact that the player may be engaged with the enemy. Therefore these behaviours are not responsive. Interruption can be supported by HSMs and BTs but behaviour resumption needs to be added in an ad-hoc fashion. Ad-hoc solutions retain the previous context in the HSM or BT, but the interrupted behaviour may be interrupted as well. An architectural solution that uses a stack could solve the problem of interrupting and resuming behaviours [45].

NPC collaboration is difficult to incorporate using HSMs or BTs. Additional code is required to synchronise behaviours if each collaborator has a separate behaviour. If collaboration is controlled by a single joint behaviour then integrating the joint behaviour with the single behaviours of the collaborating NPCs becomes arduous. Collaboration has been successfully implemented using BTs in *Halo 3* [52] However, no support is provided by the behaviour tree architecture itself.

An *AI Planner* ascertains an action sequence that accomplishes a goal. However, past actions are not retained by planning. *Killzone 2* [53] is the first action game to adopt *Hierarchical task network planning* (HTN) [54, 55]. *Real-time planning* (RT) is used in *F.E.A.R.* [56], however, resumable behaviours are not supported. Behaviours are instead re-planned and executed from the start [45].

Effective ambient behaviours are presented in *The Sims 2* [57], however, the game model cannot be easily transferred to other genres. Interaction information is broadcasted from game objects to the NPCs. These broadcasts are then responded to autonomously by each Sim, though player interference is often required to keep them on the right track [45].

RPGs such as *Neverwinter Nights* (NWN) [58] and *Morrowind* [17] make use of scripting to incorporate behaviours. However, true collaboration is not supported. In NWN, to simulate collaborative behaviours, a script is allocated to one of the NPCs that controls the actions of both. Goal-directed AI has been used in *Oblivion* [59], however, only very simplistic collaborations are supported and as in

the case of NWN and Morrowind, does not support resumable behaviours. Rather, if behaviours are interrupted, they are restarted. These systems are not generative, therefore they are not useful to non-programmers.

Façade [60] incorporates a collaborative, interruptible and resumable behaviour model. Accumulations of reactive behaviours (known as *beats*) are used by the NPCs which can be interrupted by the PC. Collaboration is handled by parallel beats. However, Façade's framework requires a lot of manual work due to the cumbersome ABL scripting language which even experienced coders find challenging. Also, *Façade* has only two NPCs situated in a small environment. Therefore their behaviour techniques will be difficult to scale to hundreds or thousands of NPCs. A mechanism that can provide behaviours of this quality to large numbers of NPCs is required but which includes minimal authoring work [45].

Chapter 3

Design

This chapter presents the proposed model to address believable behaviour of human-like NPCs, starting with a brief definition of the model and subsequently further elaborating on the main concepts that the model involves. Following this, some example characters are visualised with the concept of the model applied in order to illustrate how the model works and to showcase its potential. Afterwards, the topic of procedural content generation is discussed and thereupon examples of its use within the model are exemplified. Lastly, an important feature of the model is discussed, the purpose of which is to drive NPC behaviour.

3.1 Defining the Model

An individual's behaviour at any moment in time can result from a number of factors including current mental or physical state. Also, a key influencer is the individual's personality which can often greatly affect their mental state. With this in mind, it was decided that the model should consider two dimensions of behaviour: mental-physical state and personality. This is realised through the use of a *state vector*, an instantaneous evaluation of the NPCs mental and physical states, and a collection of *state modification vectors*, that capture how particular activities affect the mental and physical state of the NPC. A single state modification vector exists for each activity that the NPC can perform. It is important to note that the state vector and the state modification vectors are very simple items of data which do not maintain a record of any previous behaviours that the NPC may have performed.

State	State Vector
0	[0, … N]
1	
.	
.	
N	

Activity	State Modification Vector
Activity 0	[0, 1, … N]
Activity 1	[0, 1, … N]
.	.
.	.
Activity Z	[0, 1, … N]

State Modification Vector	State Vector
[a, b, … c]	[+a, +b, … +c]

FIGURE 3.1: General depiction of the model.

3.1.1 Mental-Physical State Vector

A *multidimensional* approach is used to form a state vector. This simply means that multiple independent values are used concurrently to produce an overall evaluation of the NPC's mental and physical states. This is in contrast to a one-dimensional approach which would typically amount to a basic overall description similar to "I am happy" or "I am sad" with various intervals between either state. A multidimensional approach allows for a powerful representation of mental-physical state and has the following advantages over a one-dimensional approach:

- More detailed state representation
- Greater character depth
- Greater behavioural complexity

Using multiple individual values to form a state vector, a more detailed representation of the NPC's overall state is achieved. Rather than it being defined simply as, for example, "I am sad", it can instead be defined as, for example, "I am sad" and "I am exhausted" and "I am hungry" and etc. This is a richer overall representation of the NPCs mental-physical state and portrays greater character depth since more detailed information is obtained about the NPC's current state. This then allows for enhanced behavioural complexity to be supported.

An example state vector for a game such as Skyrim [1] is illustrated in Table 3.1. Here, the state vector includes two categories of states: Mental and Physical. Mental states include mood and energy, while physical states include satedness and soberness. Through the use of *floats* for state values, a wide range of behaviours

can be supported. Also, since the logic that alters state is mathematical rather than procedural, changes and calculations can be made with code that requires little maintenance. All state values lie in the range [-1.0, +1.0].

-1.0	**State**	**+1.0**
Sad	MOOD	Happy
Exhausted	ENERGY	Rested
Hungry	SATEDNESS	Full
Drunk	SOBERNESS	Sober

TABLE 3.1: State Vector Values, Their Range and Meaning.

3.1.2 State Modification Vector

A state modification vector is, as its name suggests, a state modifier. It is n-dimensional, where n is equal to the number of states in the state vector and where any number of these states are affected. Unlike the state vector, its values are fixed and do not change once they have been set at runtime.

The number of state modification vectors required correlates with the number of activities that the NPC undertakes. An activity is any discrete action which the NPC can perform as part of its daily routine. For example, an NPC may have an activity such as “eat lunch” or “go to sleep”. Essentially for everything the NPC does, its state values will change according to the associated state modification vector. Through this, a greater depth of character is portrayed and a sense of personality is established.

3.2 Illustrating the Model

In many games, human-like NPCs appear to be personified according to their in-game character classes. In a game such as Skyrim [1], typical examples include blacksmith, farmer and hunter among others. This goes some way towards instilling in the player a sense of believable and realistic behaviour as the character appears to have a job or purpose in the game world with unique skills and abilities. However, believability soon breaks down as the NPC exhibits predictable behaviour patterns and it becomes obvious that the NPCs routine is predetermined and planned out according to a specific schedule. The NPC also shows

no sense of desire or purpose in accomplishing the tasks. Essentially, it lacks the subtleties that make the character truly believable. Without an individual personality, likes and dislikes, pastimes, social interaction and other such qualities, the NPC ultimately is not very believable. To illustrate how the model might work in Skyrim [1], some example characters have been conceptualised as follows:

3.2.1 The Drunken Farmer

A daily routine of the common farmer NPC might include a typical farming activity such as tending to the crops while intermittently going to get something to eat. Perhaps later the farmer will go to the local tavern for a while to have a few drinks and afterwards go back home to bed. A routine like this in the game is entirely plausible. However, if the model is applied with a state modification vector associated with each activity, suddenly the character becomes a lot more interesting as there is now something which appears to drive the behaviour. Take for example the state vector and state modification vectors illustrated in Figure 3.2. Here, the farming activity has a negative impact on the NPCs mood, energy and satedness states. In other words, while carrying out the farming activity the NPC becomes unhappy, exhausted and hungry. The activity of drinking on the other hand has a positive impact on mood, energy and satedness while having a negative impact on the soberness state. In other words, drinking makes the NPC happy, rested, full and drunk. This portrays the farmer as a somewhat unique individual who apparently does not like their job but does however like drinking at the local tavern. Rather than being defined simply as a "Farmer" the NPC can now be defined as "The Drunken Farmer", a much more interesting and unique character to encounter in the game.

3.2.2 The Enthusiastic Hunter

Hunters are another type of character encountered throughout the world of Skyrim, usually found in the forests, mountains and swamps. They typically wear hide armour or leather armour, boots and gloves and sometimes a hat. Generally, their weaponry consists of a hunting bow, iron arrows and an iron dagger. Their interaction with the environment around them mostly involves stalking animals such

(A)

(B)

-1.0	State	+1.0
Sad	MOOD	Happy
Exhausted	ENERGY	Rested
Hungry	SATEDNESS	Full
Drunk	SOBERNESS	Sober

(C)

Activity	State Modification Vector
Farming	[-0.8, -0.3, -0.3, +0.0]
Drinking Alcohol	[+0.8, +0.3, +0.2, -0.6]
Playing Music	[+0.5, +0.0, -0.1, +0.0]
Sleeping	[+0.3, +0.8, +0.0, +0.6]
Eating	[+0.3, +0.3, +0.8, +0.0]

FIGURE 3.2: The Drunken Farmer. (A) adapted from [61], (B) State vector, (C) Activities with associated state modification vectors.

as deer, elk and wolves. Hunters are, by default, non aggressive unless attacked. Additionally, they offer a limited merchant service, usually selling hides and meat.

The above description portrays a well defined but static character type where all hunters are essentially the same, ultimately lacking personality and other human-like qualities. Applying the concepts of the model, however, can quickly transform a hunter character into a unique individual. Take for example the hunter illustrated in Figure 3.3. The state vector and state modification vectors depict a hunter who seemingly is extremely happy when hunting and who also is somewhat happy when practicing archery, though only half as much as when hunting. One could describe this character as "The Enthusiastic Hunter". However, if the character's "sanity" state is brought into question, its personality could be entirely different. For example, an extremely negative value depicts a deranged character whose love for hunting could be borne out of the character's unstable mental well-being, which in turn will likely have a direct impact on the character's mannerisms. Such a character would be very interesting to encounter, with a level of personality which does not already exist in the game.

(A)

(B)

-1.0	State	+1.0
Insane	SANITY	Sane
Sad	MOOD	Happy
Exhausted	ENERGY	Rested
Hungry	SATEDNESS	Full

(C)

Activity	State Modification Vector
Hunting	[+0.0, +0.8, -0.1, -0.2]
Archery Training	[+0.0, +0.4, -0.1, -0.2]
Sleeping	[+0.0, +0.1, +0.8, +0.0]
Eating	[+0.0, +0.1, +0.2, +0.8]

FIGURE 3.3: The Enthusiastic Hunter. (A) adapted from [62], (B) State vector, (C) Activities with associated state modification vectors.

3.3 Procedural Content Generation

Procedural Content Generation (PCG) is often referred to as the study and development of algorithms that produce content automatically. Game *content* refers to any adjustable game component that may have an impact on player experience, with the exception of NPC behaviour. This typically includes elements such as camera profiles, levels, maps, music, narratives, quests, rulesets and terrains. The automatic generation of content in games provides several benefits. First, PCG can drastically reduce the effort and cost associated with content creation and make it simpler to adapt content to the player. Second, content can automatically alter the game to suit the meeds and predilections of individual players, maximising game re-playability. Third, PCG can formulate solutions that initially would not have been considered by the designer, therefore challenging human creativity. Today's commercial game development sees the use of PCG reach a peak of interest, showcased by successful games such as *Minecraft* [63], which is almost entirely procedurally generated [64].

As stated in Chapter 1, an objective is to support the automatic generation of NPC attributes in order to reduce the amount of manual work that the developer must complete. The model supports this in a number of ways. First is the NPC's *profession*, which is randomly chosen from a selection of in game character classes and assigned to the NPC at runtime. A profession has an associated set of activities that together form a job routine for the NPC. Second, the NPC's clothes can be selected to match the chosen profession, along with any character specific inventory items. For example, if the NPC is assigned the "farmer" profession, some typical farm clothes can be selected to be worn and items such as a woodcutter's hand axe could be added to its inventory. This same idea can be applied to all activities, even those not related to the NPC's profession. For example, if an activity such as going to the pub for a few drinks is started, a sub-activity that could be taken care of automatically might be to change clothes to something more appropriate. Finally, all state modification vector values are set once at runtime and do not change. This can be handled automatically whereby a random number, that lies in the specified range, is generated for each value. This means that the values do not have to be hard-coded, freeing up the developers time and leading to a distinct set of state modifiers with the potential to trigger diverse behaviours.

3.4 Activity Selection Manager

When considering believability, it is reasonable to assume that the actions of the NPC at any point in time can have an immediate impact on how believable or realistic the player perceives the situation to be. However, oftentimes it depends on the player's interpretation of the situation that makes it believable or not. For example, a farmer wandering about a field in the middle of the night might seem strange and at first thought could be deemed unrealistic or not very believable. However, the NPC could in fact have a valid reason for this seemingly erratic behaviour. Maybe the key to the house was dropped while tending to the crops or perhaps a few too many drinks were had in the local pub. Whatever the reason may be, the player could actually be intrigued upon encountering such, meaning that even the most unlikely behaviour cannot be considered outright as not very believable. Therefore, deciding when the NPC should perform certain actions can be quite problematic as it is hard to gauge what the player will regard as believable behaviour.

The activity selection manager is an essential feature that forms the core of the model and which ultimately drives NPC behaviour. It attempts to support believability by prioritising and selecting the most appropriate activity for the NPC to carry out through a series of complex processes.

Since every activity has an associated state modification vector which adjusts the values of the state vector, the activity selection manager can choose the most appropriate activity by examining the state vector and determining which state requires the most attention. To illustrate this, take for example the scenario of a farmer NPC performing its daily routine. After some time it is likely to get tired and hungry. When the current activity finishes, lets say chopping wood, the activity selection manager would determine that the next activity would be to go and get something to eat, rather than the next work related activity of gathering and stacking the wood blocks. In other words, a work routine should be interruptible by miscellaneous activities when appropriate. With that, there are many considerations which must be taken into account. For example, in this situation a distinction must be made between being physically exhausted, i.e., requiring something to eat, and mentally exhausted, i.e., needing to go to sleep. Of course, there is also the possibility of a profession, hunting for example, that will require the NPC to wait some time before its activities can be carried out. Therefore, other miscellaneous activities such as pastimes or social activities can be performed in the meantime. However, this adds a degree of complexity as now the activity selection manager must keep track of in game time and the duration of activities when choosing an action for the NPC to perform. Additionally, a record must be kept of the activities that have been completed in order to prevent the NPC from repeating itself.

In choosing an action, the Activity Selection Manager must consider the NPC's profession and determine at what time of the day is it most appropriate for the NPC to carry out work related activities. For example, a farmer might start its routine early in the morning whereas a bartender would likely not work until sometime in the afternoon or evening. There is also the possibility of some work related activities which could be undertaken at various times of the day. For example, hunting could be done during the day or at night.

Once the NPC's profession has been analysed and a starting time for its activities has been determined, the activity selection manager must then amass a collection

of miscellaneous activities that the NPC can perform throughout the day. Examples can range from basic necessities such as eating and sleeping to the more personal or social activities such as, for example, drinking at the local pub or playing music. It is worth noting that every activity must have a specified duration. This could be handled automatically with a random duration from a set range, or each activity could be manually given a fixed duration by the developer.

The pseudocode below outlines very simply how the Activity Selection Manager could be implemented to select an activity.

```
activity selectedActivity;

activity SelectActivity(float currentGameTime, float[] stateVector)
{
  If a state vector value is below its satisfactory threshold
    selectedActivity = activity to satisfy the
      aforementioned state
  Else
    If the current game time is within the range for the NPC's
      profession
        selectedActivity = a profession activity that has not
          been done already
    Else
        selectedActivity = a miscellaneous activity

  return selectedActivity
}
```

Chapter 4

Implementation

This chapter reviews the implementation of the proposed model to support believable behaviour of background NPCs in open world games. First, the tools and technologies used in the implementation are briefly mentioned and explained. Following this is a detailed analysis of how the tools were utilised to provide a virtual environment for the model to work with. Finally, the implementation of the model is examined.

4.1 Getting Started

Since implementing the model in the context of open world games was of primary concern, it was decided to utilise the editor used to create content for Skyrim [1], known as the Creation Kit. This modding kit includes all the tools needed to create custom content, using the game's files. It also includes its own scripting system made specifically for the Creation Kit, known as Papyrus.

4.1.1 Papyrus

Papyrus is a fully-fledged scripting language that works by receiving Events from the game and sending function calls to it. At its core, papyrus is object oriented in nature and is the key driving factor behind much of the functionality for animated objects that require interaction by the player or NPC. A papyrus script is basically

a plain-text source document (.psc file) that can be written and compiled into a form the game can interpret (.pex file) using any text editor.

4.1.2 Skyrim Script Extender

The Skyrim Script Extender, otherwise known as SKSE, is a library that extends the scripting capabilities within the Creation Kit. It does so without any adverse effects as no executable files on disk are modified. Additionally, the SKSE allows modders to "hook" into Skyrim's game engine as some mods require tight integration with it.

4.1.3 Installation

In order to get up and running with the Creation Kit, a Steam account and a copy of Skyrim are required before it can be installed. With the prerequisites acquired, it is as simple as launching and logging into Steam, navigating to the games library and selecting "Tools" from the drop-down filter. Here the "Creation Kit" can be located and chosen to be installed. Afterwards, the SKSE can be downloaded from the official download site and following the instructions in the documentation, it can be installed with relative ease.

4.2 The Creation Engine

The Creation Engine is the engine used to power Skyrim [1]

4.2.1 Data Format

A common data format utilised across many Bethesda Game Studio titles is used by the Creation Engine. Most significant are the Master files, with the ".esm" extension, which are substantial collections of data. The master file, Skyrim.esm, consists of all the base game's data. Smaller collections of data known as Plugins, with the ".esp" extension, can be loaded "on top" of the master file. Plugins are capable of modifying or referencing data held within a master file and can also incorporate wholly new data.

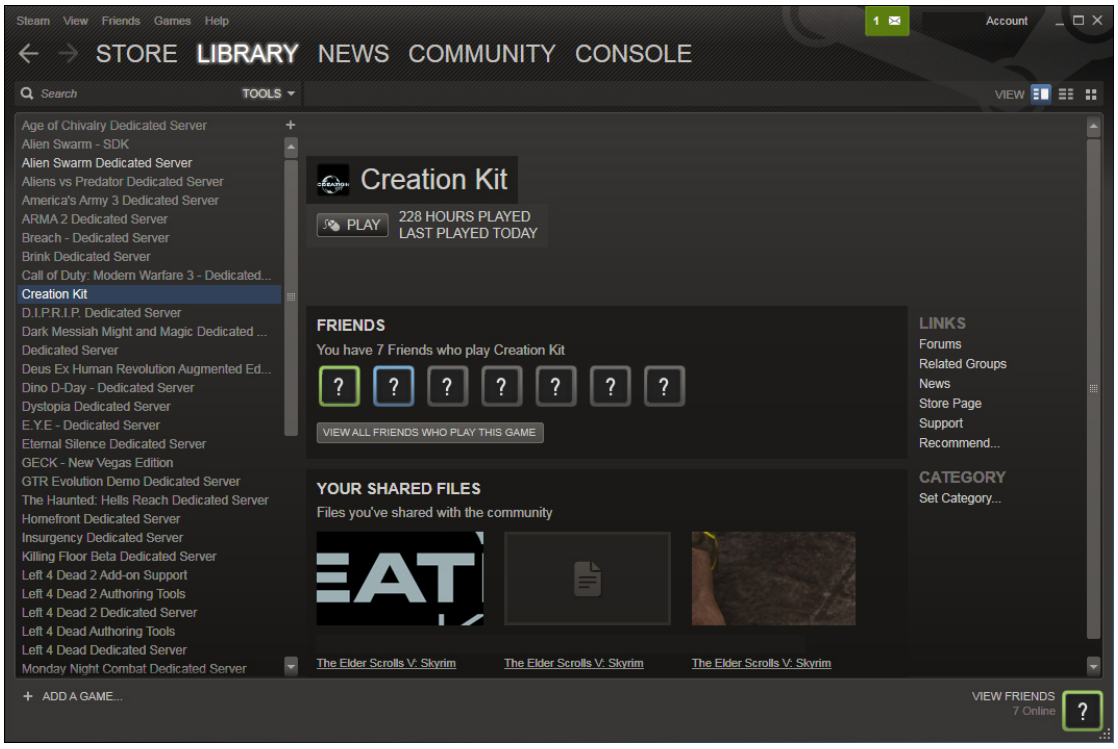


FIGURE 4.1: Locating the Creation Kit in Steam Tools [65]

4.2.2 Creating & Loading the Plugin

It is advisable to use the Skyrim.esm and Update.esm files as the plugin's master files. Initially, no data is loaded when the editor starts, therefore these files should be loaded first. Skyrim.esm takes some time to load due to the vast amount of data that it contains. When loading is finished a "plugin" can be created which acts as the mod file where all work is saved. The plugin must be loaded when the game starts in order for it to take effect. This can be done from the main game launcher, selecting "Data Files" and choosing the plugin. Alternatively, the SkyrimCustom.ini file can be opened and the plugin(s) to load can be manually specified.

4.3 Utilising the Creation Kit

To support implementation of the model, many features of the Creation Kit were utilised.

FIGURE 4.2: Skyrim Launcher

4.3.1 Creating an Actor

It was decided that to demonstrate the model in action, it would be best to showcase its implementation on a newly created NPC rather than on an existing character within the game. This would more closely correlate with the model's implementation in commercial development.

In Skyrim [1], the Actor (Class) Script defines a particular type of object known simply as an Actor. All animate beings, humanoids and creatures alike, are created as a reference to (or an instance of) this object, allowing everything about a character to be defined. Actors execute AI routines which facilitates movement around the environment, engaging in combat, interaction with numerous other types of objects such as containers, doors, activators (buttons, levers, etc.) and other actors.

The Creation Kit provides a convenient user interface to facilitate the setup and creation of a character. Figure 4.3 depicts the Actor window, as is, before the new character is created.

At first it can be overwhelming but most fields are actually quite self-explanatory including Id, Name and Short Name. One of the more essential fields is the checkbox entitled Unique. This informs the engine whether there should only be one instance of the actor in the game world. The Traits tab on the right-hand side allows fundamental characteristics to be set such as the NPC's race, sex, height, weight, voice type and more. The Inventory tab, though not depicted, allows the

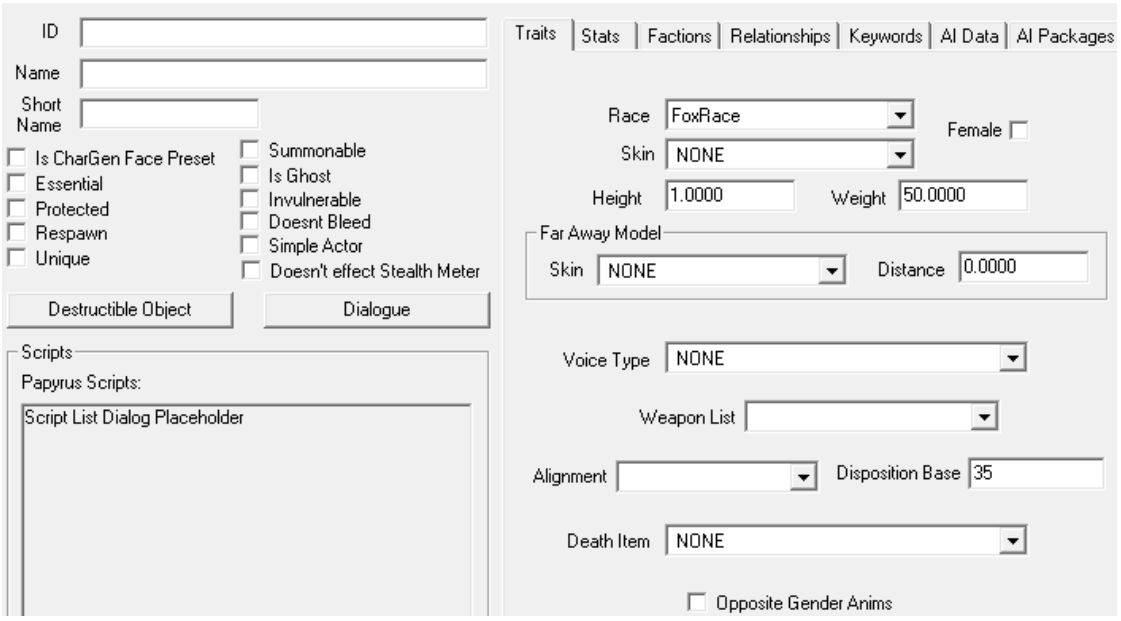


FIGURE 4.3: Creating an Actor (Screenshot taken from Actor Window)

user to set the character's outfit, in other words, what clothes the character will wear. It is important to note that much of this functionality can be manipulated in code and therefore is also the subject of procedural content generation. Figure 4.4 depicts the Actor window for the in-progress creation of a character.

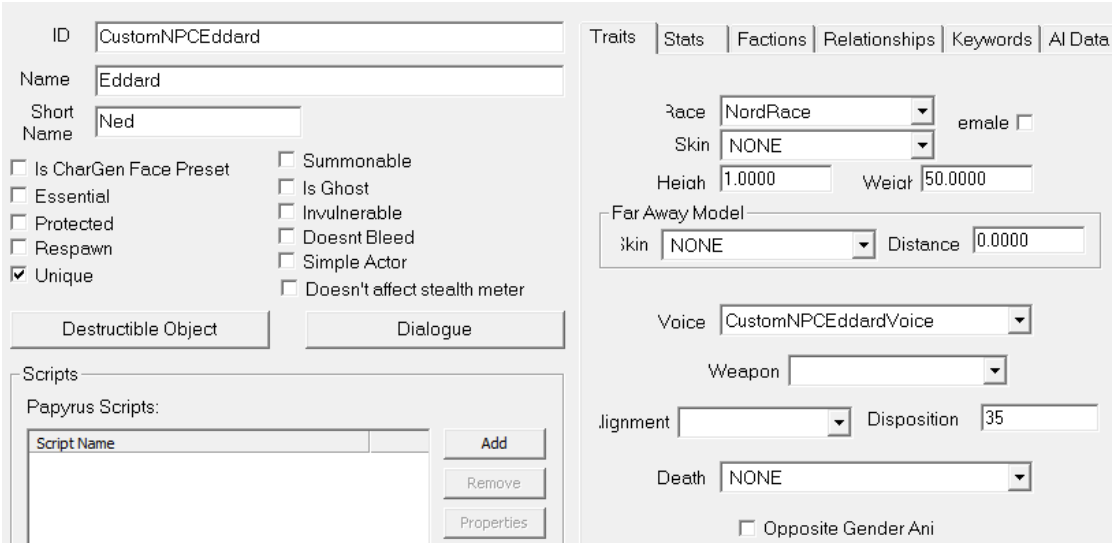


FIGURE 4.4: Defining the Actor (Screenshot taken from Actor Window)

4.3.2 Placing the Actor

Described so far has been the creation of the base object for the actor. In order to place it in the game, a reference must be created for it. First a location within

FIGURE 4.5: Placing the Actor (Screenshot taken from Render Window)

the game world must be chosen for the actor to be placed. This can be done by navigating the Cell View window and selecting the desired location from the cell list to display it in the Render window. Creating the actor reference is as simple as locating it in the Object window and dragging it into the Render window. The actor object can then be manipulated, i.e. moved and rotated, until the user is satisfied with its position in the game environment.

4.3.3 Idle Markers

Comprising of specific animations or an assortment of animations that can be chosen from randomly, idle markers allow actors to exhibit certain behaviour when they reach a particular position in the game world. However, any animation which the actor cannot perform will be ignored. Idle markers are typically used by actors when performing a Sandbox procedure which essentially allows the actor to "roam" around searching for idle markers to use, which provides the illusion of interaction with the world. Figure 4.6 depicts some farming idle markers which can be used by an actor to perform a range of farming related animations.

FIGURE 4.6: Idle Markers (Screenshot taken from Render Window)

4.3.4 Furniture

All furniture objects in the game, such as chairs and beds, can be used by actors. Tables with benches attached are classified as furniture objects, however, tables with no benches are not because actors do not have any associated animations. In Skyrim [1], Furniture Objects (orange markers) are used to implement many idle animations instead of the traditional Idle Marker Objects (blue 'I' markers). While furniture is more prohibitive since animations are fixed to the object and cannot be altered, the advantage lies in the more accurate animation poses that they provide. Furniture objects can also be used to "anchor" an actor in a certain animation or pose using the Set Restrained function, even when it would normally terminate. Furniture marked with the "Special" keyword allows the actor to use objects such as alchemy tables, arcane enchanters, bar stools, forges, grindstones etc., which all have specific animations for the actor to play.

4.3.5 XMarkers and XMarker-Headings

XMarkers are generic markers used to designate a specific position in the game world.

FIGURE 4.7: Bar Stool - Special Furniture (Screenshot taken from Render Window)

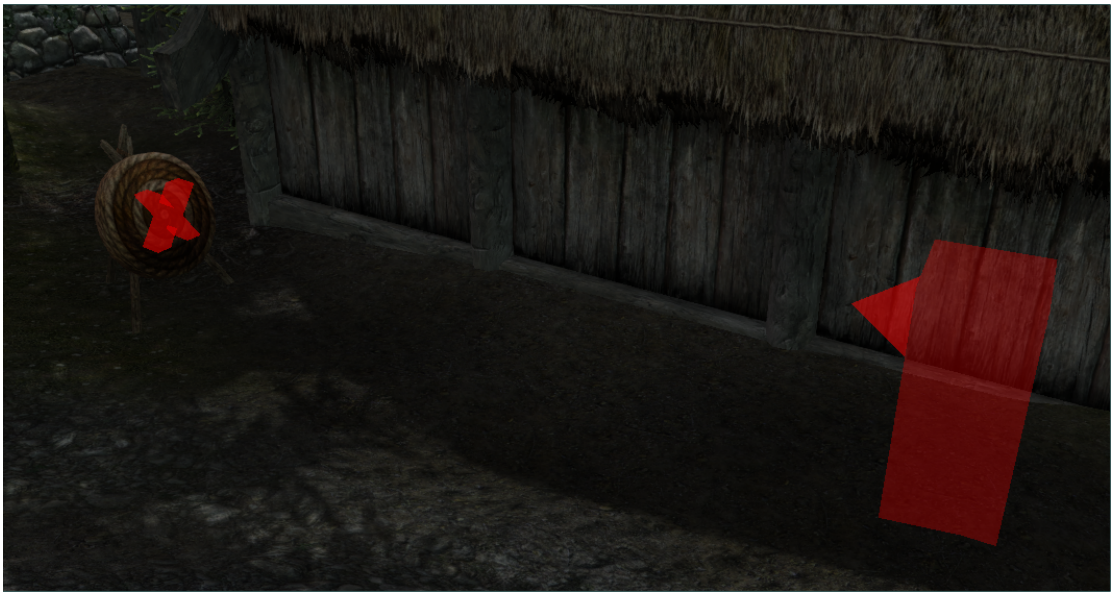


FIGURE 4.8: XMarker & XMarker-Heading (Screenshot taken from Render Window)

XMarkers can be used, for example, to mark the location for which an actor must travel to, as illustrated in Figure 4.7. They can also be used to mark an object which the actor can engage in combat with, as depicted in Figure 4.8.

XMarkerHeadings are generic markers that have a directional arrow. They are used to designate a certain position of the game world, for example, a patrol point or, as portrayed in Figure 4.8, a position from which to engage in combat.

4.3.6 Creating the Activities

In Skyrim [1], a routine is generally created for NPC's by assigning the actor various AI Packages as depicted in Figure 4.9.

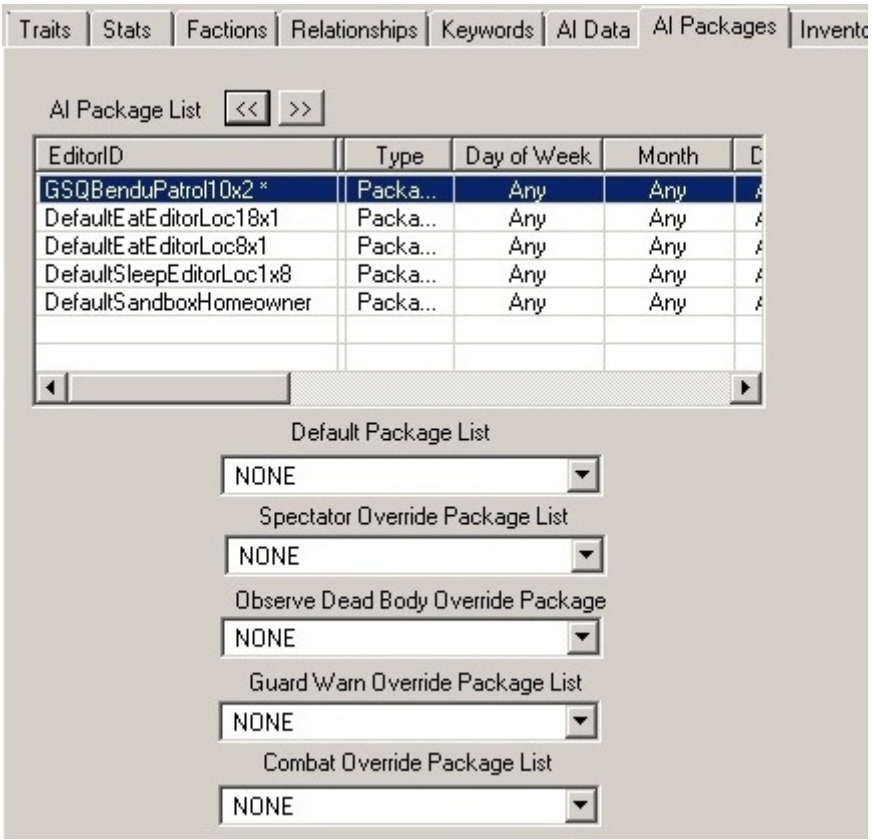


FIGURE 4.9: AI Packages (Adapted from [66])

Packages are the primary way of controlling an actor's behaviour. Essentially all actors have a Package Stack, which is a stack of packages that exist on an actor reference. Each package corresponds to a particular behaviour that will be performed under specific conditions. A package is generally an instance of a Package

Template, a prebuilt Procedure Tree which presents standardised, reusable functionality for common behaviours. The data of a procedure tree can be modified, however, its structure and logic cannot. A procedure tree is a stack of Branches ("Nodes") consisting of Procedures, the fundamental components of a package which represent the individual actions that form the behaviour.

A package allows data inputs to be defined which are passed as parameters to the procedures at run time to control the behaviour. For example, the "Destination" data of a "Travel" package, which represents a location to travel to, can be defined. It could be set, for instance, to a particular XMarker reference or the actors "Editor Location". Now take multiple procedures contained within a procedure tree and a complex series of actions that form the behaviour becomes more apparent. Analysing the package tree structure can depict the "logical thinking" of an actor as it performs the specified actions. For example, the "Sit" package template defines a package that first tells the NPC to "Find" a chair. If one is found, the NPC "Sits" in the chair, however, if one is not found, the NPC is instructed to "Wander" around instead.

While the package system is quite powerful, the conventional appliance of packages to provide a routine for NPCs has its drawbacks. Defining a list of specific behaviours to perform at a precise time according to a schedule means that the behaviours are predetermined, leading to predictable behaviour patterns. The proposed model aims to implement NPC routines in a more dynamic fashion through the Activity Selection Manager as described previously in Chapter 3. Essentially the model requires the behaviours to be completely decoupled and as generic as possible. In other words, behaviours should no longer be amassed together as a single collection of AI packages and their use should not be restricted by a schedule. In order to achieve this a different approach must be taken.

It became apparent that within the Creation Kit, Quests (stories or adventures in Skyrim) have a significant feature known as an Alias. This is an id which can be assigned to things used by the quest, such as actors and locations. Data items (AI packages, scripts, dialogue, etc.) can also be linked to the alias rather than to an explicit object in the game world. It was decided to apply this concept to the formulation of a NPC's routine. An individual behaviour can be created as a Quest, which for the purposes of this project is referred to as an Activity. An alias within this quest can be assigned to the NPC in question and an AI package that controls the specific behaviour can be created and linked to the alias. Since

quests can be manipulated in code, for example, it can be instructed to start or stop, an individual behaviour now exists that can be controlled independently.

To illustrate the formation of an activity, lets take for example a typical farming task. Figure 4.10 depicts the creation of a "Woodcutting" activity. As described previously, an activity is first created with an appropriate id and name. For now this is all that is required. It is important to note, however, that some functionality is enabled by default. The "Start game enabled" option instructs the activity to start immediately when the game loads, while the "Run once" option ensures the activity is only ever run a single time. This is undesirable and thus should be disabled.

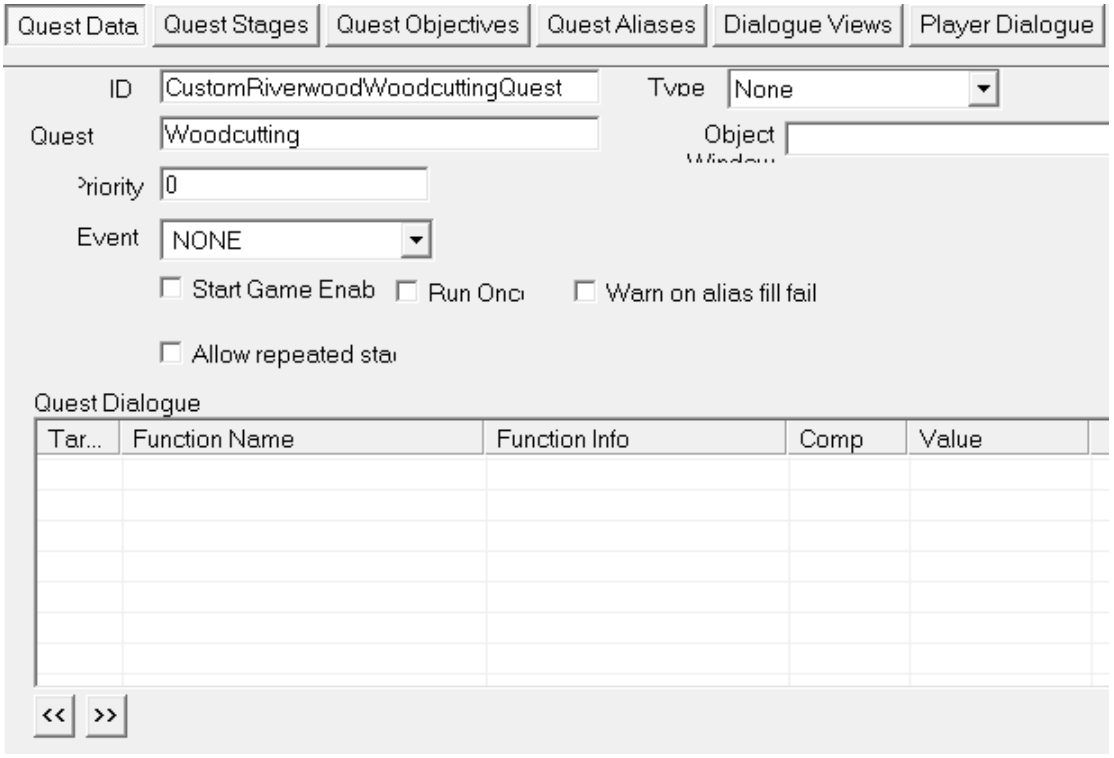


FIGURE 4.10: Creating an Activity

The next step is to create an alias via the Alias tab in the activity's main toolbar. The alias is given an appropriate name and the NPC is lined to the alias by selecting the "Unique actor" option. Lastly the AI package which controls the "woodcutting" behaviour is linked to the alias through the "Alias Package Data" category. Figure 4.11 portrays the final alias product for the "Woodcutting" activity.

Of course, in order to assign the alias a specific AI package, it must first be created. As illustrated in Figure 4.12, a new package is established consisting of a procedure

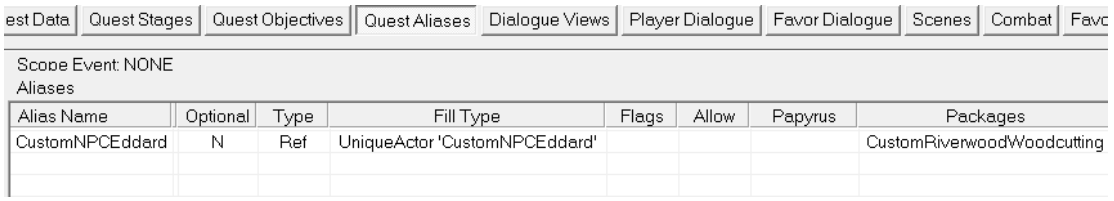


FIGURE 4.11: Woodcutting Activity Aliases

tree that ultimately enables the NPC to perform the "Woodcutting" activity. The NPC is initially instructed to "travel" to a location, which is specified as the area near the wood chopping block where it will perform the activity. The NPC is instructed to unlock any doors it may encounter to reach its destination. After this the NPC is then instructed to "Sandbox" when it reaches its destination. The Sandbox procedure enables the NPC to perform specific animations associated with any idle marker within a specified radius. In this case, the radius is limited to a small range ensuring the NPC will only use the wood chopping block, which is essentially an idle marker that only allows an actor to perform the wood chopping animations. While sandboxing, a Wait procedure is activated simultaneously. Generally this procedure is used to stop the animations of an actor for a specified amount of time, however it can also be used as a timer for the Sandbox procedure which itself never ends. By setting the wait timer, disabling the function to stop the actors animations and instructing the entire package to complete when the Wait procedure ends, a package is generated to enable the NPC to specifically perform the "Woodcutting" activity.

4.4 Implementing the Model

With a new NPC created and a range of behaviours ready to be used, the implementation of the model could now be attempted. A concept was formulated whereby a Papyrus script would be attached to the NPC to drive its behaviour. In other words, all aspects of the model including the state vector, state modification vector and activity selection manager would be implemented according to Papyrus's coding conventions.

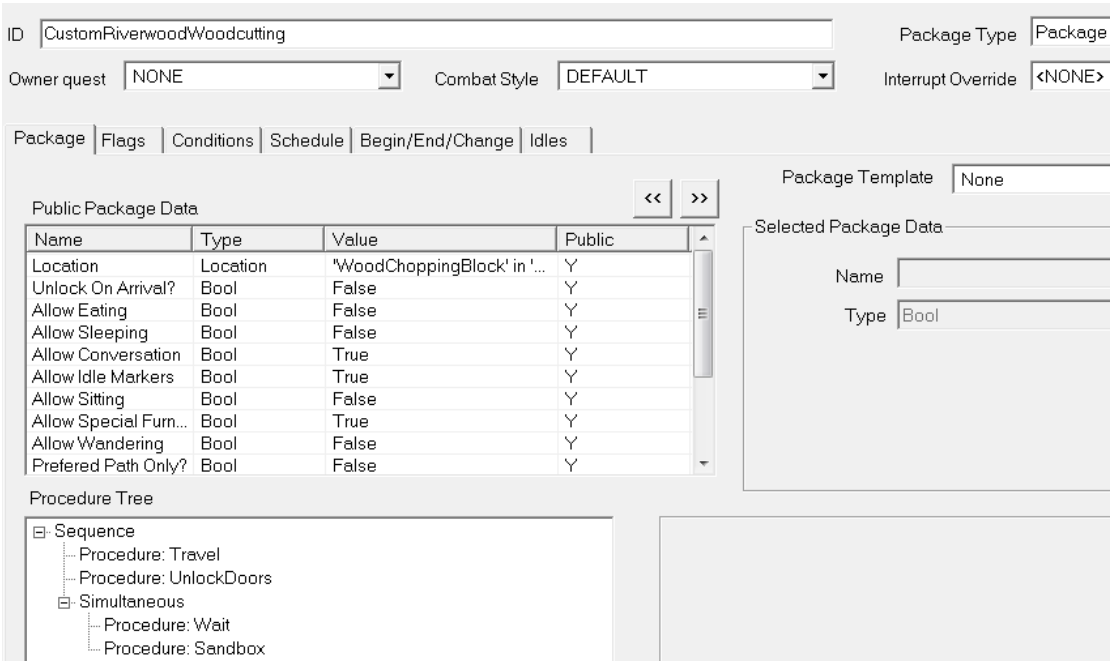


FIGURE 4.12: Alias AI Package

4.4.1 Creating the Script

As mentioned earlier, Papyrus scripts can be edited and compiled using a text editor such as Notepad++ or Sublime Text. Before the script could be written, however, it had to be decided what type of script it would be. In other words, what object type is it intended to be executed on. Once this was determined, the script could then be created by declaring it at the top of a .psc file and extending the object script that it is based on. Since this project involves running the script on NPCs, it must extend the Actor script.

4.4.2 Writing the Script

With the script created, the main body of the script could be written. First, the "Import" keyword is utilised to allow global functions to be used within the script without prefixing them with the name of the script that they belong to. Once the necessary libraries were imported, the state vector and state modification vectors were declared as simple float arrays.

Properties, which are essentially variables that other scripts can access and control how they are used, are declared to hold various objects such as a specific actor, outfit or activity. Papyrus supports three kinds of properties, the full property,

the auto property and the auto read-only property. The full property provides complete control over what the property does and what it may or may not allow. The functions within the property definition are regular functions, only they must be a "Get" or "Set" function. Either may be omitted, however, at least one must be present. An auto property is essentially a public variable that automatically creates a hidden variable along with hidden "get" and "set" functions to manipulate it. An auto read-only property is a publicly accessible data item which cannot be altered. It must be initialised with a value when declared, for obvious reasons. The auto property was used in this project.

```
; ACTOR
Actor Property CustomNPC Auto

; OUTFITS
Outfit Property AFarmerOutfit Auto

; ACTIVITIES
Quest Property Woodcutting Auto
Quest Property ArcheryPractice Auto
```

An Event is a special type of function which the game calls whenever something happens. The *OnInit* event will run when the script is initialized. It is within this event that the state vector and state modification vectors are initialized and assigned values. It was decided to assign all state vector values the neutral value of 0. All state modification vector values on the other hand, were assigned values using a function that returns a random value between -1 and +1. Also, an array to store all activities is initialised and populated and a function call is made to the activity selection manager in order to start the NPC's routine.

As discussed earlier in Chapter 3, a full implementation of the activity selection manager involves many complex processes and considerations. For this prototype implementation of the model it was decided to demonstrate its potential with a simplified version, whereby activities would be selected at random rather than according to a specific mental or physical state. It is worth noting that due to constraints and limitations of the Papyrus language the implementation of the activity selection manager is somewhat crude as Papyrus does not support multidimensional arrays. In order to associate a state modification vector with each individual activity, the indexes of the two separate arrays had to be matched. The activity selection manager chooses a random index value and uses it to set the current activity and state vector.

Since the state modification vectors can increase or decrease the values of the state vector, measures to restrict the values had to be implemented in order to ensure they continually lie within the specified range. This was achieved by using a simple function to "bound" the state vector values.

```
float Function Bound(float value)
    if (value > 1.0)
        value = 1.0
    endIf

    if (value < -1.0)
        value = -1.0
    endIf

    return value
EndFunction
```

Other than the initial function call in the *OnInit* event, the activity selection manager is called whenever an activity finishes. This is handled using the *OnPackageEnd* event which is called when an actor finishes an AI package. Since each activity comprises a single package, the event can be used to determine when the next activity should be selected.

```
Event OnPackageEnd(Package akOldPackage)
    Debug.Notification("AI Package Ended...")
    SelectActivity()
EndEvent
```

For debugging purposes, the state vector and state modification vector values needed to be displayed on screen within the game. This is handled with the use of the *OnActivate* event which is called when the player selects the NPC in the game, which displays a message box with the appropriate information. This may be cumbersome if the ability to converse with the NPC is enabled which uses the event. However, it is sufficient for the purposes of this prototype implementation.

Something of note is that the idea of different NPC professions has been omitted from the implementation. This is due to the issues that arise in the activity selection manager when determining what activities the NPC can and cannot perform.

Chapter 5

Evaluation

In Chapter 1, a number of objectives were defined, outlining the requirements for the model to succeed in supporting believable behaviour of background characters in the context of open world games. This chapter presents an evaluation of each of these objectives to determine whether the model succeeds and if it could enable the videogame industry to deliver even more engaging and immersive gameplay experiences.

5.1 Improving On Existing Approaches

The implementation demonstrates a promising model with much potential. Through the use of the State Vector and State Modification Vectors a sense of personality of the character is gained, creating unique individuals that would be more interesting and exciting to encounter within the game world. However, the objective to offer a tangible improvement over existing approaches has been only partially satisfied. It remains to be seen if the model truly does so, with a more sophisticated implementation of the Activity Selection Manager likely needed to determine such. Currently, activities are selected at random rather than according to a specific state, meaning that a realistic and believable NPC routine is not entirely possible. The algorithm outlined in Chapter 3 illustrates how to support the improved selection of activities, however, due to time constraints it was not implemented in this prototype of the model. Therefore, there is no definitive proof just yet that the defined model can be used by the video game industry to deliver more engaging

and immersive gameplay experiences. Future work is necessary to procure more conclusive evidence.

5.2 Supporting Procedural Content Generation

The prototype implementation of the model somewhat satisfies the objective of supporting procedural content generation to reduce developer workload, through the random generation of state modification vector values. However, in its current form, it is not very efficient. There is currently no limitation on the range of values that can be selected, leading to adverse effects and scenarios that are not very believable. For example, an activity such as "eating" should always have a positive modifier value for the "satedness" state. Indeed, it could have a positive value between a specified range so that it neither falls too low nor reaches too high for it to be deemed unrealistic. If state vector values lie in the range [-1.0, +1.0], perhaps an appropriate range for a modifier that affects the "satedness" state would be [+0.3, +0.7]. The range for modifier values typically depends on the state that it affects. For the above example of the "eating" activity, it is not appropriate for the modifier that affects the "satedness" state to be a negative value. However, for the modifier that affects the "mood" state, a range of [-1.0, +1.0] would be valid.

The algorithm that has been conceptualised to enhance the functionality of the Activity Selection Manager also provides further support for procedural content generation. If an appropriate activity can always be chosen for the NPC to carry out at any moment in time, the developer's workload will be reduced as there is no manual implementation work involved.

5.3 Efficiency of Implementation

To test whether the model lends itself to efficient implementation in order to accommodate CPU and memory usage, a series of complexity analyses can be carried out. For example, a complexity analysis could be carried out on the code of the entire implementation or on the code that forms the Activity Selection Manager, assuming of course a more complete implementation has been accomplished.

For this prototype implementation of the model, a simple efficiency test is undertaken to determine mathematically the number of variables required for each NPC and the number of variables required for each profession.

Given:

a = the number of activities
s = the number of values in a state vector (i.e. the number of states)
n = the number of values in a state modification vector = s
p = the NPC's profession = 1

$$\text{The total number of state modification vector values} = a \times n$$
$$= a \times s$$

$$\therefore \text{The number of variables for each NPC} = (a \times s) + \text{s} + \text{p}$$
$$= (a \times s) + \text{s} + 1$$

Each profession can have any number of activities associated with it. From a worst case scenario perspective (in terms of complexity), the number of variables for each profession equals the total number of activities.

$\therefore$ The number of variables for each profession = a

5.4 Impediments

While the prototype implementation goes some way towards supporting believable behaviour of background characters in open world games, there are a number of impediments that ultimately hold it back from realising its full potential. Since a great deal of time was spent learning how to utilise the Creation Kit editor and its scripting system, Papyrus, the model has not been implemented in full. Consequently, the Activity Selection Manager remains only partially implemented. As mentioned earlier, this results in NPC routines that are not entirely believable. Further work is required to see this feature reach its full potential.

As it stands, the model has been implemented as a script which conforms to the coding standards of the Papyrus scripting language. This is somewhat inhibitory

as Papyrus lacks support for some important data structures such as multidimensional arrays. It was later discovered midway through development that the Skyrim Script Extender, SKSE, not only adds further support for Papyrus scripting, but also enables mods to be coded in C++ and integrated much more tightly with the game engine. In hindsight, this would have been the ideal method of implementation.

There has been some unforeseen consequences as a result of utilising the scripting system to implement the model. Seemingly when a script is attached to a NPC, it is re-run when the player loads the game save file. The ramifications of this being that the NPC's routine could be interrupted and anything randomly generated will be reassigned different values which could cause adverse behavioural affects. Incorporating the model into the game engine via the SKSE would solve this problem.

Chapter 6

Conclusion

By way of conclusion this chapter will detail some contributions made by the work undertaken in this dissertation, potential future work and some final thoughts on the project.

6.1 Contributions

While there is much future work remaining, the work completed to date has made several contributions.

6.1.1 Supporting Believable Behaviour of NPCs

As mentioned earlier in Chapter 1, researchers have worked on many alternatives to the typical industry methods used to create believable interactive characters. Following in the footsteps of other AI researchers who have implemented models of emotion and personality, this dissertation supports believable behaviour of NPCs through the formation of a new model based on mental-physical state and personality. By choosing activities for the character to carry out based on the its current state, and modifying the states for every activity that must be accomplished, a sense of personality of the character is gained, forming a unique individual that would be interesting to encounter in a game.

6.1.2 Showcasing Potential

With a prototype of the model implemented in Skyrim [1], its potential is showcased through visual representations of its various components. While the model is not implemented in full, it justifiably demonstrates the use of the State Vector, State Modification Vectors and the Activity Selection Manager that which together help support believable behaviour in NPCs.

6.1.3 Highlighting the Impediments of Existing Approaches

As a result of a prototype of the model implemented in Skyrim [1], the impediments of the existing approach to character behaviour used within the game become quite apparent. To illustrate, take the example of how AI packages are utilised. Essentially, behaviours are compiled into a list and allocated specific scheduling, meaning that behaviours are predetermined and ultimately very predictable. Applying the concept of the model, however, results in behaviours that have no specific scheduling. Instead, behaviours would be selected based the NPC's current state, their profession or a random miscellaneous activity. Ultimately the model makes for less predictable and more believable NPC routines.

6.2 Future Work

Taking into account the impediments of the current implementation of the model, future work could be undertaken to reimplement the model at a much lower level in C++ using the Skyrim Script Extender, SKSE. This would provide greater control over implementation details, equipping the developer with the means to integrate the model directly into the game engine.

To determine whether an improvement over existing approaches to character behaviour is attainable, the model will have to be realised in full. Therefore, future work involves building upon the basic version of the Activity Selection Manager implemented for this prototype and seeing it reach its full potential. The pseudocode outlined in Chapter 3 provides a starting point to accomplish this.

In section 5.2, a problem is highlighted whereby the randomness of the State Modification Vector values leads to adverse effects and scenarios that are not

very believable. The solution to limit the values within a specified range could be implemented in a similar fashion to the "bound" function used on the State Vector.

6.3 Final Thoughts

This dissertation has shown how a model to support believable behaviour of background characters in the context of open world games has the potential to offer something more than that of existing approaches. While the implementation might be lacking in some areas, it is sufficient for a first prototype to demonstrate the concept. Any deficiencies can be attributed to the steep learning curve of the Creation Kit and its scripting system. With some future work, it is quite plausible that the proposed model, or a variation of, could be used by the videogame industry to deliver even more engaging and immersive gameplay experiences in open world games.

After spending much time with the Creation Kit, it has become apparent that it supports many of the traditional techniques used in game AI such as finite state machines, scripting and pathfinding. With additional support through the SKSE, it is questionable whether the Creation Kit could be used in academia as an educational tool. This might be worth looking into, however, the steep learning curve must be taken into account.

Appendix A

Appendix

Attached is a CD with the plugin (.esp) file, the source script (.psc) file, the compiled script (.pex) file and any other relevant assets used for this dissertation project.

References

[1] Bethesda Game Studios. The Elder Scrolls V: Skyrim. [DVD, Blu-ray Disc, Download], 2011.

[2] Henrik Warpefelt and Björn Stråat. A method for comparing npc social ability. In *Computer Games and Allied Technologies 2012*. Global Science and Technology Forum, 2012.

[3] Magnus Johansson, Mirjam P. Eladhari, and Harko Verhagen. Complexity at the cost of control in game design? 2012.

[4] Magnus Johansson, Harko Verhagen, and Mirjam P Eladhari. Model of social believable NPCs for teacher training: Using Second Life. In *Computer Games (CGAMES), 2011 16th International Conference on*, pages 270–274. IEEE, 2011.

[5] Henrik Warpefelt and Björn Stråat. Breaking immersion by creating social unbelievabilty. In *Accepted for presentation at the SOCIAL.PATH track of AISB 2013*, 2013.

[6] Henrik Warpefelt and Björn Stråat. The dos and donts of npc social capability. 2013.

[7] Henrik Warpefelt and Björn Stråat. Anti-heuristics for maintaining immersion through believable non-player characters. 2013.

[8] Edward Castronova. *Synthetic Worlds: The Business and Culture of Online Games*. University Of Chicago Press, 2005. ISBN 0226096262.

[9] Richard Bartle. *Designing Virtual Worlds*. New Riders Games, 2003. ISBN 0131018167.

[10] Robert Purchese. Warren Spector: Hey Carmack, Sweeney, stop rendering and start making believable AI. http://www.eurogamer.net/articles/2012-08-15-warren-spector-hey-carmack-sweeney-stop-rendering-and-start-making-believable-ai, August 2012. Online; Accessed July 18, 2013.

[11] S. T. Coleridge. *Biographia literaria*. 1817.

[12] David Woodruff Smith. Phenomenology. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Fall 2011 edition, 2011.

[13] A Bryan Loyall. *Believable agents: building interactive personalities*. PhD thesis, Stanford University, 1997.

[14] Joseph Bates. The role of emotion in believable agents. *Commun. ACM*, 37 (7):122–125, July 1994. ISSN 0001-0782. doi: 10.1145/176789.176803. URL http://doi.acm.org/10.1145/176789.176803.

[15] Joseph Bates. The Nature of Characters in Interactive Worlds and The Oz Project. 1992.

[16] Valve Corporation. Half Life 2. [Optical disc, Download], 2004.

[17] Bethesda Game Studios. The Elder Scrolls III: Morrowind. [CD-ROM, DVD, Download], 2002.

[18] Irrational Games. Bioshock Infinite. [Optical disc, Download], 2013.

[19] Wesley Yin-Poole. Bioshock Infinite's Elizabeth: Ken Levine on creating the best AI companion since Half-Life 2's Alyx Vance. http://www.eurogamer.net/articles/2012-12-17-bioshock-infinites-elizabeth-ken-levine-on-creating-the-best-ai-companion-since-half-life-2s-alyx-vance, December 2012. Online; Accessed July 25, 2013.

[20] uesp.net. Fargoth. http://images.uesp.net/a/a9/MW-npc-Fargoth.jpg, . Online; Accessed July 18, 2013.

[21] elderscrolls.wikia.com. Lydia. http://elderscrolls.wikia.com/wiki/Follower_%28Skyrim%29?file=Lydia.png. Online; Accessed July 18, 2013.

[22] Rockstar. Red Dead Redemption. [Optical disc, Download], 2010.

[23] rockstargames.com. Bonnie macfarlane. `http://media.rockstargames.com/rockstargames/img/global/downloads/wallpapers/games/reddeadredemption_bonniemacfarlane_1920x1080.jpg`. Online; Accessed July 18, 2013.

[24] wikipedia.org. Alyx vance. `http://en.wikipedia.org/wiki/File:Alyx_Vance.png`. Online; Accessed July 18, 2013.

[25] bioshock.wikia.com. Elizabeth. `http://bioshock.wikia.com/wiki/Elizabeth?file=Elizabeth_R1.png`. Online; Accessed July 18, 2013.

[26] Mike Williams. Grand Theft Auto franchise hits 125 million shipped. `http://www.gamesindustry.biz/articles/2012-11-28-grand-theft-auto-franchise-hits-125-million-shipped`, November 2012. Online; Accessed July 18, 2013.

[27] Mike Rose. Skyrim Ships 10M Worldwide, 'Fastest Selling Title In Steam's History'. `http://gamasutra.com/view/news/128736/Skyrim_Ships_10M_Worldwide_Fastest_Selling_Title_In_Steams_History.php`, December 2011. Online; Accessed July 18, 2013.

[28] Ernest Adams. *Fundamentals of game design*. New Riders, 2010.

[29] Magy Seif El-Nasr, Leslie Bishko, Vernica Zammitto, Michael Nixon, Huaxin Wei, and V Athanasios. *Handbook of Digital Media in Entertainment and Arts*, chapter 22. Believable Characters. New Riders Games, 2009.

[30] Komarine Romdenh-Romluc. Merleau-ponty and the power to reckon with the possible. *Reading Merleau-Ponty: On Phenomenology of Perception, edited by T. Baldwin. Abingdon: Routledge*, 2007.

[31] Maurice M Ponty. *The Structure of Behaviour*, page 168. Beacon Press, 1963.

[32] Hubert L Dreyfus. A merleau-pontyian critique of husserl's and searle's representationalist accounts of action. In *Proceedings of the Aristotelian Society (Hardback)*, volume 100, pages 287–302. Wiley Online Library, 2000.

[33] Magnus Johansson and Harko Verhagen. Where Is My Mind-The Evolution of NPCs In Online Worlds. In *J. Filipe and A.L.N. Fred (Eds.), ICAART 2011 - Proceedings of the 3rd International Conference on Agents and Artificial Intelligence.*, volume 2, pages 359–364. SciTePress, 2011.

[34] Harko Verhagen, Mirjam Eladhari, and Magnus Johansson. Social believable NPCs: a conceptual model and analysis of current NPC models. In *ECREA digital games preconference*, 2012.

[35] Mei Yii Lim, João Dias, Ruth Aylett, and Ana Paiva. Creating adaptive affective autonomous NPCs. *Autonomous Agents and Multi-Agent Systems*, 24(2):287–311, 2012.

[36] Wander Jager. *Modelling consumer behaviour*. PhD thesis, University of Groningen, 2000.

[37] Wander Jager, MA Janssen, HJM De Vries, J De Greef, and CAJ Vlek. Behaviour in commons dilemmas: Homo economicus and Homo psychologicus in an ecological-economic model. *Ecological economics*, 35(3):357–379, 2000.

[38] Wander Jager and Marco Janssen. An updated conceptual framework for integrated modeling of human decision making: The Consumat II.

[39] Kathleen Carley and Allen Newell. The nature of the social agent*. *Journal of mathematical sociology*, 19(4):221–262, 1994.

[40] Tetsunari Inamura, Tomohiro Shibata, Hideaki Sena, Takashi Hashimoto, Nobuyuki Kawai, Takahiro Miyashita, Yoshiki Sakurai, Masahiro Shimizu, Mihoko Otake, Koh Hosoda, et al. Simulator platform that enables social interaction simulation - SIGVerse: SocioIntelliGenesis simulator. In *System Integration (SII), 2010 IEEE/SICE International Symposium on*, pages 212–217. IEEE, 2010.

[41] Raúl Arrabales, Agapito Ledezma, and Araceli Sanchis. Towards conscious-like behavior in computer game characters. In *Computational Intelligence and Games, 2009. CIG 2009. IEEE Symposium on*, pages 217–224. IEEE, 2009.

[42] Jerry Ball, Christopher Myers, Andrea Heiberg, Nancy J Cooke, Michael Matessa, Mary Freiman, and Stuart Rodgers. The synthetic teammate project. *Computational and Mathematical Organization Theory*, 16(3):271–299, 2010.

[43] Kiran Ijaz, Anton Bogdanovych, and Simeon Simoff. Enhancing the believability of embodied conversational agents through environment-, self-and interaction-awareness. In *Australasian Computer Science Conference (ACSC 2011). CRPIT*, volume 113. Citeseer, 2011.

[44] Helder Coelho and AR Costa. On the intelligence of moral agency. *Proceedings of the Encontro Português de Inteligência Artificial (EPIA 2009), New Trends in Artificial Intelligence, Aveiro, Portugal, October*, pages 12–15, 2009.

[45] Maria Cutumisu and Duane Szafron. An architecture for game behavior ai: Behavior multi-queues. In *Proceedings of the 5th Artificial Intelligence for Interactive Digital Entertainment Conference (AAAI)*, pages 20–29, 2009.

[46] Daniel Fu and Ryan Houlette. *The Ultimate Guide to FSMs in Games*, volume 2 of *AI Game Programming Wisdom*, pages 283–302. Charles River Media, Massachusetts, USA, first edition, 2004.

[47] David Harel. Statecharts: A visual formalism for complex systems. *Science of computer programming*, 8(3):231–274, 1987.

[48] Ubisoft. Assassin's Creed. [Optical disc, Download], 2007.

[49] Damian Isla. Handling complexity in the Halo 2 AI. In *Game Developers Conference*, volume 12, 2005.

[50] Microsoft Game Studios. Halo. `http://www.halowaypoint.com/en-us/games`, May 2013. Online; Accessed July 18, 2013.

[51] Valve Corporation. Left 4 Dead. [Optical disc, Download], 2008.

[52] Bungie. Halo 3. [Optical disc], 2007.

[53] Guerrilla Games. Killzone 2. [Blu-ray Disc], 2009.

[54] Earl D. Sacerdoti. The nonlinear nature of plans. In *Proceedings of the 4th international joint conference on Artificial intelligence - Volume 1*, IJCAI'75, pages 206–214, San Francisco, CA, USA, 1975. Morgan Kaufmann Publishers Inc. URL `http://dl.acm.org/citation.cfm?id=1624626.1624657`.

[55] A Champandard, T Verweij, and R Straatman. The AI for Killzone 2's multiplayer bots. In *Proceedings of Game Developers Conference*, 2009.

[56] J. Orkin. Three States and a Plan: The AI of F.E.A.R. In *Proceedings of the Game Developer's Conference (GDC)*, 2006.

[57] Maxis. The Sims 2. [CD, DVD, Direct Download], 2004.

[58] Bioware. Neverwinter Nights. [CD-ROMs], 2002.

[59] Bethesda Game Studios. The Elder Scrolls IV: Oblivion. [DVD, Blu-ray Disc, Download], 2006.

[60] Michael Mateas and Andrew Stern. Façade: An Experiment in Building a Fully-Realized Interactive Drama. In *Game Developers Conference, Game Design track*, volume 2, page 82, 2003.

[61] giantbomb.com. Addvild. `http://www.giantbomb.com/images/1300-2348931/`. Online; Accessed July 20, 2013.

[62] uesp.net. Hunter. `http://images.uesp.net/2/21/SR-npc-Hunter.jpg`, . Online; Accessed July 26, 2013.

[63] Mojang. Minecraft. [Download, in-browser], 2011.

[64] Geogios N Yannakakis. Game ai revisited. In *Proceedings of the 9th conference on Computing Frontiers*, pages 285–292. ACM, 2012.

[65] creationkit.com. Steam tools. `http://www.creationkit.com/images/0/02/SteamCreationKit.jpg`, . Online; Accessed August 06, 2013.

[66] creationkit.com. Creation kit ai packages. `http://www.creationkit.com/images/3/31/Package_Tutorial_NewPackage6.jpg`, . Online; Accessed August 09, 2013.