An Emotional Model For Background Characters In Open World Games

by

Niall Mullally, B.A. PCAM

Dissertation

Presented to the

University of Dublin, Trinity College

in fulfillment

of the requirements

for the Degree of

Master of Science in Computer Science

(Interactive Entertainment Technology)

University of Dublin, Trinity College

September 2014

Declaration

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.

Niall Mullally

September 1, 2014

Permission to Lend and/or Copy

Acknowledgments

Firstly I would like to thank my supervisor Dr. Mads Haahr for all his support and help throughout this dissertation.

I would also like to thank Tiarnán McNulty for his help and allowing me to demonstrate the model in a world environment he created.

Niall Mullally

University of Dublin, Trinity College
September 2014

An Emotional Model For Background Characters In Open World Games

Niall Mullally
University of Dublin, Trinity College, 2014

Supervisor: Dr. Mads Haahr

The goal of this dissertation is the creation of an Emotional Model that can be used to increase believability of background Non-Player Characters (NPCs) in open world games. Background AI used in these games are simplistic due to the fact that they must be very efficient and not hinder the performance of the game, as the number of NPCs visible on the screen at any time could be 30-40+. While these NPCs are simplistic they are vital for the immersion of the player into this new world. Created in this dissertation is a emotional model which can be easily applied to NPCs to provide them with emotional states. An Action Selection Manager evaluates the NPC's current emotional state and chooses the next action based on these values. The actions picked alter the emotional state of the NPCs. Each NPC's personality comes from being randomly assigned Traits. These traits, e.g. Energetic or Alcoholic, give each NPC a unique personality. Traits are assigned at the initialization of the NPC and modify its actions. The model is tested and evaluated in a game environment and is deemed successful since the NPCs created are plausible. This model could have potential use in games with further improvements.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The topic of this dissertation is the design and creation of an *Emotional Model* that will give *Non-Player-Characters* (NPCs) emotions, and allow them to make choices based on their emotions. It is intended for friendly background NPCs in vast Open-World games. In general these background characters in open world games have very limited personality or uniqueness due to constraints of memory and efficiency. Typical behaviour for these NPCs are static predictable daily schedules such as wake up, go to work for a few hours, go to the pub, then sleep and repeat. While this is plausible for a wanderer such as the player in the game as he generally only passes by towns or stays for short durations, on closer inspection it can ruin the immersion of the environment to see these NPCs with little variations. This dissertation proposes an emotional model that will change the NPCs behaviour based on an emotional aspect leading to a more realistic and unpredictable *Artificial Intelligence* (AI) whose choices will depend on their current emotional state.

1.1 Open World Games

"Open World Games" meaning has changed radically over the last decade in the games industry. Most game environments tended to be small for most games with linear maps and limited NPCs. These worlds would be inhabited by NPCs which would generally have the same behaviour but different dialogue. In recent years most games are developed to be open world due to pressure from the player's desire for a more dynamic and non linear world, giving the player an option and not forcing him down a singular path.

Open World Games such as the Grand Theft Auto [1] series with their recent title *Grand Theft Auto 5* [2] puts the player in a massive metropolis. Just like a normal city it is populated with pedestrians walking around like real life. These pedestrians show the illusion that they are living a daily life such as eating or interacting with objects in the world. However, these AI in GTA5 are generated when the player is within a certain area and they interact with the environment around them to give the impression that they are just going about their business. It makes the player feel like he is in a real world that is populated with people. However, in games such as GTA5, the NPCs are not constant, if the player moves to far they will de-spawn and be gone forever. This is due to the massive amount of memory that would be required to save all these NPCs. While the player does not form a bond or opinion with these NPCs they fulfil their role perfectly in giving the impression to the player that it is a living world with people.

In this dissertation the meaning of "Open World Games" is used to refer to games like *Skyrim* [3]. Skyrim is a *Role-Playing Game*(RPG) where the player is given an identity and is dropped into a new world. The game has a massive continuous world that is only loaded once and that the player can explore freely. This world is filled with many villages that are populated with villagers. These villagers each have a certain pre-designed personality such as a blacksmith or fisher and are constant throughout the world.

The design for this emotional model is for a game like this, where immersing the player into the environment is vital for game play and with a little bit of extra work the believability of these NPCs could be drastically increased.

1.2 Motivation

Similar to the use of music and sound as a main method to add immersion in movies, RPGs primarily rely on the environment and NPC behaviour, as well as these for immersion. In open world RPGs such as Skyrim or *S.T.A.L.K.E.R* [4], immersion is vital for the game experience. This immersion can be lost when interacting with with a NPC that has a predictable behaviour. Since these vast worlds are inhabited by hundreds of NPCs that are an essential part of the believability of the world, their role is vital for drawing the player into this new world. If the believability of these NPCs is lost this could damage the player's reaction to the world.

With advances in computational power the ability to create large scale environments becomes trivial, in the gaming industry many recent games are becoming large open world games to give the player more control rather than a linear map that guides the player through.

One area which could potentially benefit from this is Massively Multi-player Online(MMO) games which over the last decade have exploded, games such as *World Of Warcraft* [5] (currently 8M - 10M active subscribers), *Guild Wars 2* [6] (3M-4M), *Second Life* [7] (800k subscribers) and many other games (*Aion* [8], *RuneScape* [9], *Lineage II* [10]).

In MMOs most of the immersion comes from interacting with other player's rather than the NPCs. That is why they can afford to use simplistic AI behaviour with the exception of bosses or enemies that the player's will frequently be against. The friendly NPCs that inhabit the world lack personalities and are just used as a wanted board to give the player a task or quest and provide some information. The only variations are through dialogue or appearances and their schedules are generally scripted. Increasing the believability of these NPCs could greatly add to the player's immersion in the game and make him feel like he is actually part of a living environment where the NPCs will not be at the same corner all day and night, but rather they would be going about their own business trying to fulfil their own desires and goals. The downfall of this would be more effort on the player's part, having to chase down the NPC before being able to complete a task.

In single player open world games where there is only one player who is in a new environment, the actions of the NPCs will have a more prominent role in immersing the player to this new world.

The current focus in single player open world games is primarily the content, such as dungeons or quests that the player can do and the detailed environment, these two parts of the game alone takes a huge amount of resources to develop. For some AAA games which push the boundaries on graphics there is very little room to implement a move advanced AI model which could potentially take up resources. This is unfortunate but has been occurring for a long time.

Throughout the history of gaming AI progression has been left behind compared to graphics which has made huge improvements. Old games used to focus mainly on the player and have very simple AI, most commonly using a Finite State Machine and A* for path-finding with some dialogue depending on what state the agent is in. This is still the case in some games created as a lot of games can afford to neglect the AI and focus on other aspects of the game. This is unfortunate as games rely on AI heavily at times since the player always encounters AI.

Since early games such as *Pac-Man* [11], AI have been vital for the game play experience. Pac-Man is a straightforward game, the player must navigate a maze and collect pellets to get points, but what makes it interesting is the chase from the AI. In Pac-Man each ghost has its own individual "personality" in how they attack or approach Pac-Man [12].

As current generation consoles and computers become increasingly more powerful there is room to improve these AI characters. In recent years AI characters are becoming much more important as they are the tools that tell the story or shape the story of the game [13]. This is why the characters telling the storys or part of the story must appear intelligent to not break the immersion for the player.

These AI advances are mainly seen in single player First Person Shooters (FPS) where there is not as much detail as a large environment and the number of AI is limited. These AI characters are used for telling dynamic and intriguing stories and also to form a bond with the player. One example which is hailed as one of the most important AI in game history is Alyx from *Half-Life 2* [14], another more recent and more advanced example is *BioShock Infinite* [15]. In BioShock Infinite the player has a companion called Elizabeth who has been secluded from normal life and ends up being taken across the world by the player. Throughout the journey she will often interact with random objects, make observations and also help the player by finding objects the player misses. These AI are mainly used to draw the player into the game, and help them throughout the story by giving information.

Telltale Games recently developed *The Walking Dead* [16] and *The World Among Us* [17] both episodic games use a sort of point and click mechanic for the player as he is lead on a story. One of the main features in these games is the AI. Based on conversations and decisions you make in the game, you can hurt your relationship with that AI which effects the outcome of the story.

While these advanced AI models would be ideal for a large game like Skyrim, the reason they can afford these complex models is that they are tailored towards one specific character. Another difference is the number of NPCs that could potentially be visible on the screen is 30-40+ so the models used must be efficient and non complex to not hinder the performance.

In Skyrim the NPCs have their own assigned houses in villages or cities. There is no random NPC creation other than bandits that attempt to steal from the player or other dynamic events that the player encounters on the roads.

Unpredictable agent behaviour is somewhat desirable to not only keep the player interested in the game but also as a challenge that unpredictability may bring. If the player notices a pattern it will become obvious and can potentially be exploited, such as a villager that will leave his house to drink at 8pm every night then the player can potentially steal from this villager after he leaves his house knowing his exact pattern. The player may also notice that in every village the AI are mirrored from other villages with minor changes such as dialogue. This would unravel the illusion of the world if the player notices that all the villages through the game are similar.

This is the main motivation behind this dissertation, trying to improve the fidelity of these background characters to try give each of them a unique personality. To enable the NPCs to decide for themselves what their next action should be based on their current emotional levels. This will ultimately lead to less predictable NPCs. This would mean that on the player's journey through the world it would be very unlikely to meet two characters that act in the same way. There could be similarities but these would be result of the character's characteristic rather than pre designation.

1.3 Objectives

The main objectives for this dissertation are listed below. The overall goal is the design and creation of an *Emotional Model* which is used to increase the believability and can be easily applied to games. This will be a contribution of the goals listed below. The main objective is *Increasing Believability* in these background characters. The model is to be able support *Procedurally Generated Characters* and also support the ability for the developer to *Customize NPCs* for their liking. The final goal is create an *Efficient Model.* These goals are achieved through creation of the emotional model.

The emotional model is what will be created to achieve increased believability of NPCs in games. If the model is to be successful, it is desirable to create a generic model that can easily be applied to other games. The model will also support the ability for the developers if required to alter traits and stats for a given NPC in order to create their own unique NPCs if required.

1.3.1 Increasing Believability

As mentioned in the Motivation section, increasing believability of NPCs is the primary goal. The ability to create a model where the NPCs will have their own personality and will decide their own actions based on their current emotional state.

1.3.2 Procedurally Generated Characters

The idea is that these NPCs can be randomly generated and have random personalities assigned to them without having the developers to manually each NPC. This gives the developers more time for other aspects of the game.

1.3.3 Support Customizable Characters

While it is beneficial that the model supports procedurally created NPCs, in the best interest in keeping the model as open as possible, it should give the developer the option of creating their own custom NPCs if they require a specific one.

1.3.4 Efficient Model

The last objective is an efficient model. Since open world games have such vast view distances, and the number of NPCs currently visible on screen could potentially be 30-40+ it is vital that the model be as efficient as possible, so that it does not impact the performance of the game. Without this objective the model would be worthless and never implemented in games.

1.4 Dissertation Roadmap

Chapter 2 is focused on reviewing the current state of the art in academia and what is currently used in the gaming industry. Furthermore it tries to give a little perspective in the two areas as they differ due to different purposes. Chapter 3 is a discussion of the design of the emotional model for increasing believability in NPCs. Chapter 4 is a walk-through of the implementation process and outlines the decisions and reasoning picked during the implementation of the model. After this is chapter 5 which attempts to evaluate the emotional model and compare what has been achieved with what objectives were set at the start and results of the emotional model. Finally, chapter 6 is a conclusion of the dissertation noting what was achieved and potential future work.

Chapter 2

State Of The Art

The purpose of this chapter is to provide some insight into the relevant areas and work done on emotional models and increasing the believability in NPCs. First a discussion of believability is presented. This is followed by some work that has been done on Artificial Intelligence (AI), emotional models and believability for NPCs in the research area. These are some models that are created to simulate human behaviour in life situations such as a shopping mall [18] or how humans behave in a group environment [19]. A brief mention of a new conference which aims to bridge the gap between academia and industry is talked about in section 2.2.1.

Following this is a breakdown of some techniques used in the gaming industry to try create more believable NPCs with the use of decision making algorithms or path finding. Finally a closer look is taken at some games which are known for their advanced use of AI and what is currently used in open world games.

2.1 Believability

The primary concern of this dissertation is to make NPCs more believable through their actions. The idea is to let the decision making of these NPCs be a result of their emotions.

2.1.1 Believable Agents

The question of "What defines a believable agent?" has many different answers depending on the person, it could be the agents physical appearance, the dialogue the agent uses, the actions the agent takes or a combination of all these.

Bryan Loyall in a paper titled "Believable Agents: Building Interactive Personalities" [20] outlines what he believes are the qualities of a believable agent. He states that "A Character is considered believable if it allows the audience to suspend their disbelief and if it provides a convincing portrayal of the personality they expect or come to expect" [20][p1]. He also mentions that believable agents differ from believable characters in that they are computer based and interactive. He states that the developer of a believable agent cannot know all the situations the agent will encounter and that for the agent to be believable it must have a personality and cannot depend on the developers knowledge in every situation. Loyall lists in detail the requirements for what he believes is a believable agent.

Personality
Considered as the most important requirement and defined as "all of the particular details - especially details of behaviour, thought and emotion - that together define the individual". Each agent should have their own personality so that no two agents should ever do something the same way.

Emotion
These agents must appear to have emotional reactions and be able to express these emotions in a manner that is true to their personality.

Self Motivation
Agents should be self-motivated and act on their own accord rather than only reacting to stimuli. A more believably agent can be perceived when the agent's actions are not because of stimuli but because of the agent's own desires.

Change
That agents grow and change is believed to be a crucial step for a believable agent. Through the use of a learning algorithm or some other automatic mechanism, the agent could grow tired of eating fish and begin to dislike it after eating fish for

all of his life.

Social Relationships
: Since agents interact with other agents or the player it is important that these interactions are influenced based on the relationship between the agents and that they may also change the relationship between the agents. A difficulty mentioned is that agents declared as “friends” or “enemies” are not detailed enough and detailed behaviour and interactions that show the relationship are also needed.

Consistency of Expression
: All agents have different ways of expression, such as facial expressions, animations, movement and voice acting. In order to be believable at every moment, all of these ways must work together and to convey a unified message that is appropriate for the agent’s current personality of feelings. Inconsistency between these can cause disbelief in the believability of the agent.

Illusion of Life
: The final requirement for believability is a combination of requirements. First the agent must appear to have a goal or a purpose in life. They must have the ability to perform “multiple actions and pursue multiple higher level activities concurrently”. Agents must be “reactive” and “responsive” to environmental stimuli and these reactions must appear human like; being too fast or slow breaks down the illusion. Agents should appear “situated” in the sense that they change what they are doing and how they do it in response to the unfolding situation. Agents should have appropriate “physical and mental limits” to appear believable. They must “exist in a social context”, “broadly capable” and “well integrated”, transitions between states must appear fluid rather than abrupt.

2.2 Emotional Models In Academia

While the focus of this dissertation is primarily creating an emotional model for use in a game, techniques from academia have been integrated for the use in games. Mentioned below are some research papers on different techniques to implement an emotional model into a game. Research on emotions in computers in the area of affective computing originated with Rosalinds and Picards paper "Affective Computing" [21]. The motivation for the research is the ability to simulate empathy, where the machine should interpret the emotional state of humans and adapt its behaviour to them.

2.2.1 Social Believability In Games

Social believability in games is a new conference which aims to bridge the gap between researchers and game developers for making believable NPCs for player enjoyment and immersion. Focus has been on other aspects of AI such as path-finding in order to make a more challenging opponent for the player by given the agent the ability to use the environment for cover etc. rather than social aspect of intelligence of the agent, and while most AI in games is simply designed with the aim of fighting the player, this is starting to change with a shift from just enemies to cooperative agents that help the player (see 2.5.1). A major part that has been neglected in most AI is intelligent AI behaviour, an AI that will dynamically adjust based on several factors, and not one that has one or two pre-programmed states. The following statement is from the website [22].

This workshop aims to address this deficiency by putting forward demonstrations of work in the integration of these three aspects of intelligent behaviour, as well as models and theories that can be used for the emotional and social aspects, and for the integration between the three aspects.

This conference which shows promise for interesting papers in the future was only started in 2013. One paper presented last year by Dimas and Prada [23] presents a model called the *Dynamic Identity Model for Agents* (DIMA) with the idea to provide agents with a dynamic identity which is determined by the social context. Focus on the identity for believable and socially intelligent agents by many researchers have considered the identity as non dynamic. They believe that the identity should be dynamic and change based on the social situation the agent is in. The agent's perception about its social context will have an impact on the agent's identity which will determine the agent's decisions. The paper mentions mention the use of it in an game such as a RPG, an NPC is an elf who has her own traits, but she also has traits due to her race, and because she is a elf she has a distrust for dwarfs. This means she would act differently towards dwarfs than humans for example. The paper mentions this can become complex in RPG as the social categorizations are vast and believe that the future agents in these worlds must be aware of their surroundings and social groups in order to increase believability.

Another paper by Bruijnes [24] aims at creating a police interview training game with the use of AI, showing the potential applications that believable AI could have not just in entertainment games but their application for use in serious games as well.

2.2.2 Modelling Of The Emotional State

The modelling of the emotional states of an agent has been researched extensively, some examples are Rosalind & Picard [25] who proposed a model for the affective anticipatory reward which could be used for enhancing the efficacy of learning and decision making. Gershenson [26] proposes a model for emotions using multidimensional logic for the degree of contradiction that emotions have.

Prada [27] attempts to create believable agents with human behaviour by developing a model of personality inspired by the Five Factor Model[28].

Five Factor Model

In psychology the *Five Factor Model* (FFM) is a model for the 5 dimensions of personality that are used to describe human personality. These five factors form the acronym OCEAN.

Openness
Appreciation for art, emotion, curiosity.

Conscientiousness
Tendency to be organized and dependable, self discipline, prefer planned rather than spontaneous behaviour.

Extraversion

Positive emotions, energetic, sociability and tendency to seek stimulation in company of others.

Agreeableness
Friendly, compassionate and cooperative.

Neuroticism
Tendency to experience unpleasant emotions easily like anger, anxiety, depression and vulnerability.

While this model is extensively used for humans and psychology it is interesting to try applying this to a NPC, it is the logical next step in order to try making NPCs more human like.

Prada [27] presents a model of personality that is based on the FFM of personality for the behaviour of social autonomous agents that interact in teamwork scenarios. Their model is inspired by the *Synthetic Group Dynamics Model.* The SGD Model was created to embed social intelligence in autonomous agents that interact in small teams. It implements behaviour patterns from results of social sciences that allow agents to generate human-like group behaviours [27]. This SGD model was extended with a motivational system to support the agent's decision making.

Fuzzy Controllers

Fuzzy logic is a multiple valued logic, rather than binary sets of true or false, fuzzy logic has a range of values between 0 and 1. Because of this range of values, it is ideal for use in order to map emotions. Research on making believable NPCs has been through the use of fuzzy controllers. Acampora and Ferraguto [29] introduces a novel kind of fuzzy controller. Research areas such as psychology, sociology and education are motivated by computer game characters [30]. The work in the paper "Synthesizing bots emotional behaviours through fuzzy cognitive processes" [29] introduce a hybrid architecture for NPCs and takes into account emotions and personality to improve their human likeness through the use of a fuzzy controller approach called *Timed Automata Fuzzy Controllers* (TAFCs) enabling a qualitative and time dependant modelling of game NPCs [31]. This inference engine is used with theories from psychological areas to provide NPCs with human-like behaviour. To test this model Acampora and Ferraguto apply it to control NPCs in Unreal Tournament 2004 [32]. Their model is a two layer architecture. *Timed and Emotional Artificial Mind for NPCS* (TEAM) This higher layer is an abstract collection of artificial components capable of synthesizing cognitive processes which usually allow intelligent entities to take autonomous decisions by analysing environmental stimuli as well as their own personality, emotions and physical conditions. To apply TEAM they require a lower layer called "UnrealTEAM" which deal with the emotions, events and personalities of typical NPCs and is directly interfaced with design tools such as the UnrealEngine. TAFCs can be exploited with Orthony, Clore and Collins [33] and OCEAN psychological theories to define a novel framework for emotional behaviours representations TEAM. TEAM uses OCC & OCEAN in order to compute the emotional state and then exploits TAFCs to determine actions in a fuzzy and time dependent way.

The TEAM architecture is shown in figure 2.1 with the following parts. *Input interface*, *Emotional Model*, *Timed Decisional Model*, *Communication framework* and *Output interface.* Input receives stimuli and sends them to the emotional model and timed decision model. The emotional model updates the emotional state of the TEAM model from the input and takes into account other TEAM models emotional state. The TDM is a decision making model which uses emotions to select the next appropriate action to try maximize both the goals achievement and human likeness property of the modelled intelligent entity. TAFCs are proposed to provide an instrument to model the Timed Emotional Model of TEAM architecture.

To evaluate their model on the impact of time, emotions and communication on human-likeness of unreal bots they ran a series of experiments. One experiment is that they create 8 variants of bots, these range from bots that are able to perceive time, feel emotion and communicate to not being able to. Some results they obtained was in terms of human-likeness the best bots are ones that have emotion. They noticed the time factor only acts in a positive way if they have the ability to communicate.

The TEAM framework ultimately attempts to model NPC behaviours through simultaneously taking emotions and time into account. The OCC & OCEAN models are used to assess attributes of an agent like emotions, personality and physical condition. The TAFCs are ultimately responsible for the actual decision making.

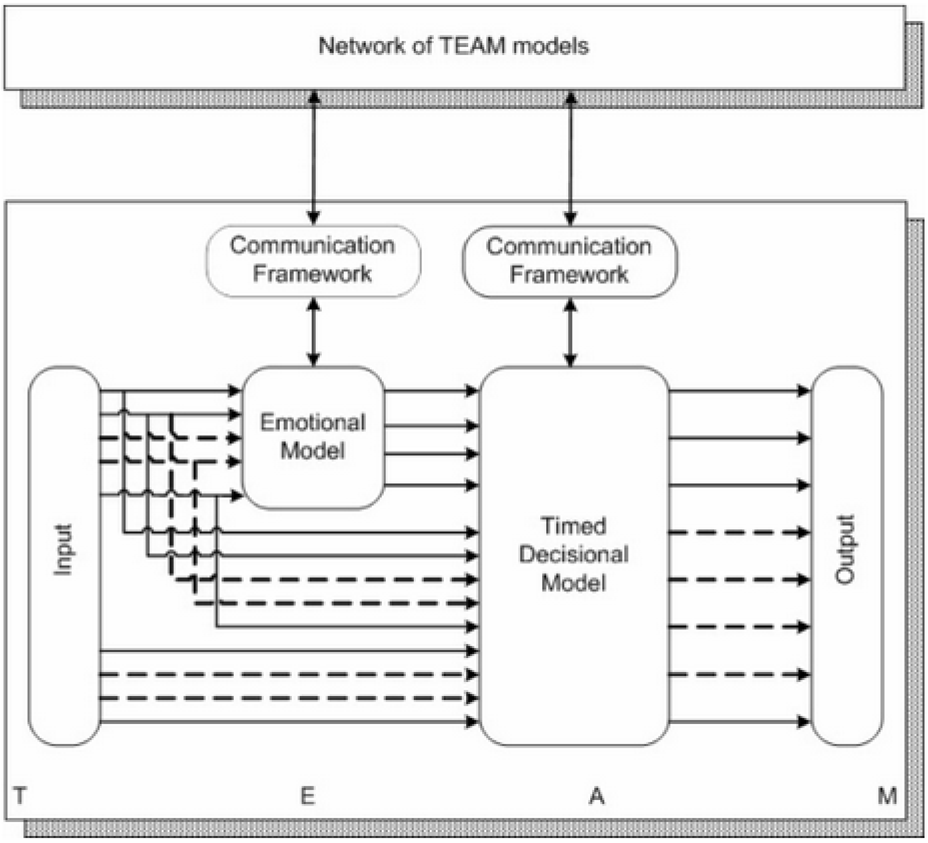


Figure 2.1: TEAM Architecture

Another paper on the use of fuzzy controllers is a paper *Psychologically Grounded Emotion Model for a NPC Fuzzy Controller* [34] which presents a psychologically grounded, CER-Based(Conditioned Emotional Response) model of emotion as a function of stimulus. The model which builds on top of a *Millenson Model.* In 1967 Millenson proposed a 3D model of Emotional Intensity [35]. Each axis represents an emotional type. with the X being anger, Y with anxiety and the Z with pleasure. Figure 2.2 shows two fuzzy inputs to yield a fuzzy output.

They use *Matlab Fuzzy Inferencing System Editor* (FISE) [36] to generate the fuzzy implementation of the Millenson Model of emotion. They have also adapted this Millenson Model to a Java application that represents the predator-prey model.

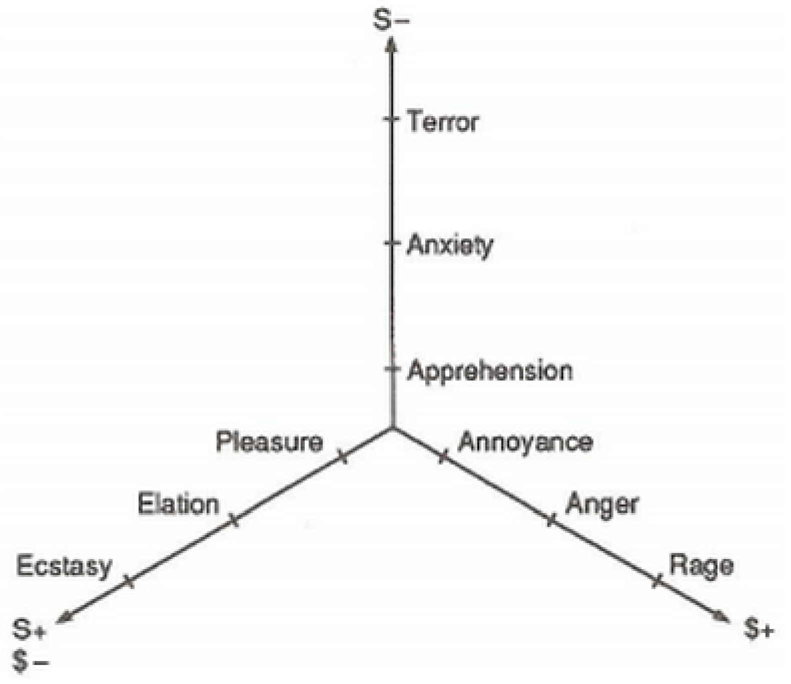


Figure 2.2: 3D Emotional Axis

Social Agents and Avatars

We were fortunate at the start of our Masters year, to be given a seminar by Prof. Hannes Vilhjalmsson on Building a Social Engine: Automating Human Territoriality for Avatars and Agents. He demonstrated CADIA Populus [37] which is a tool for a recent model for social behaviour of agents. It is part of the *Humanoid Agents in Social Game Environments* (HASGE) [38] research project at the *Centre for Analysis and Design of Intelligent Agents* (CADIA) [39], and is designed for believable autonomous behaviour of agents and avatars in massive multi-player game environments. While this model is not about decision making, it is an interesting step towards believable characters. Vilhjlmsson & Pedica [19] outline their model and demonstrates its use. In most games NPCs have no concept of personal space, and the player is free to just run up right next to them and they will act natural. In this model the NPCs will react to the players position, and if they are conversing they will try to face the player, in group situations they will readjust and reform a circle so that they are equally spaced and not have an awkward situation where one NPC is behind another. Their focus is the social positioning of the NPCs and orientation while interacting with others in social situations.

2.3 Believability In Games

The following sections deals with the current decision making techniques that are employed in the gaming industry, a review of some of the top AI in games is then presented followed by some games that are known for their advance use of AI.

Games have evolved so much over the years, from simple text games like *Zork* [40], to 2D games such as *Pac-Man* [11] and eventually 3D games. With recent technological advances, computation power has increased dramatically. Most of this power has been utilized for graphics, making games appear more realistic with highly detailed models and worlds, while AI in games has been somewhat left behind in comparison. Currently most games use some sort of generic *Finite State Machine* (2.4.1) for the AI giving them a number of states like Attack, Flee or Patrol making their reactions very predictable and use A* or similar for path-finding. Both of these methods are very common in the industry due them being solid and relatively easy to implement, however, they have their drawbacks. Recent years has seen improvements and newer techniques to replace these techniques for the AI. While AI in games has improved its only in a handful of games that the AI component really sticks out or use a variation of these techniques. An early example are stealth based games such as *Thief: The Dark Project* [41] where the AI uses an early sensory model in which the NPCs are able to react to changes in light and sound in a surrounding area. *Black & White* [42] which allows the player to teach a creature actions and to be good or evil. *Fable II* [43] where the villagers have an emotional alignment with the player depending on his actions.

Believability for NPCs is vital for immersion in games. Plenty of research has been done in this field on trying to make believable agents. Umarov & Mozgovoy [44] Outlines their method for creating a believable and effective AI agent in a virtual world through the use of learning behavioural patterns from observation and using subsequent automatic selection of winning acting strategies. Moulin [18] outlines a multi-agent-based simulation which simulates human behaviours in spatial environments such as a virtual shopping mall.

Believability in games has numerous problems compared to other areas such as animated movies because these characters are interactive. As mentioned in section 2.1.1 Loyall [20] states that the developer cannot know all the situations the agent will encounter. There are plenty of ways to try improve NPCs to make them act more realistic and believable, however, these two concepts can be thought of as separate for example voice acting, animation, detailed models, textures are all contribute to the realism of a NPC and how he looks while path-finding and decision making contribute to the agent's believability since believability only requires that the behaviour appears to be human like.

Path-Finding

Path-finding is important because if every NPC uses the same way-points, then their motion will be identical. A video [45] of a NPC in Skyrim shows a player's companion in a dungeon where there is a booby trap in the centre of an arch, which also happens to correspond to a way-point for the agent. The agent when trying to enter from one area to the next will trigger this, which causes a mechanism that pushes the agent backwards, the agent then gets up, tries to follow the player into the next area and hits the trap again, the AI is stuck in this loop and is unable to move 1 foot to the left or right to avoid this trap. Viewing such a small bug in a game leads breaks the illusion of believability. In industry common path-finding methods would be the A* approach [46] which is relatively easy to implement or the use of a navigation mesh [47] or a way-point graph where level creators manually place way-points for the AI to use. For open world games, the number of way-points could be huge depending on the world and requires developers to manually place them which could be time consuming to ensure that they work properly and not result in a problem like mentioned above. Navigation meshes cut down the number of points required for an area [48].

"Affective Decision making in Artificial intelligence" [49] introduces a new and very interesting concept, a new kind of path-finding they call emotional path-finding using emotion maps which is inspired by influence maps. Since humans can develop an emotional attachment to locations. As a result the agent takes a different path based on it's emotions and using this emotion map their path through the environment is as emotionally pleasant as possible. This creates a more believable agent and adds to the realism of the agent's path-finding especially in games with such beautiful or frightening worlds.

Decision Making

Having an agent make decisions in a human like manner is important and has been researched using models such as *Finite State Machines* or *Goal-Oriented Action Planner*, however, most of these models lack emotion as part of the decision making logic. The idea behind this dissertation is to give NPCs the ability to decide their next best action based on their current emotional state so rather than them working from 9am to 5pm without change, they will work until they need to satisfy another desire, such as rest because they are tired or drink because of thirst and then return to work if required.

2.4 Decision Making Architectures

Decision making for AI is a major area in artificial intelligence. Believability of an NPC is down to the decisions it makes, and an NPC can be considered believable if their decisions are the same that a human would make in a similar situation. There are several decision making algorithms that are used in industry, some listed below, these algorithms must be fast enough for real time use. A NPC must be able to react to events that occur in the world without hesitation, however, if the NPC only reacts we lose believability, the NPC must be able to plan ahead with the use of a planner and also able to react to external situations such as getting attacked. Currently models used in games rarely incorporate emotions in the decision making process. Behaviours changes are only due to a simple condition at times rather than a complex way using emotions that would better represent humans.

Mentioned below is a small note on some different decision making architectures used in games.

2.4.1 Finite State Machines

FSMs are extremely common in the game industry. This is because they can be relatively easy to implement and easy to comprehend. Some problems associated with FSMs is that in larger projects they can become less consistent and with growth they can become difficult to maintain. Another major problem with them is that the transitions between states are instantaneous. However, in FSMs past states are not saved which can be a problem if we wanted to transition to the previous state, a *Stack-based State Machine* can be used to correct this. A disadvantage of FSMs for believability is that transitions from state to state can be easily predicted and then exploited, if the player knows these states, such as a certain enemies attack pattern, fighting that type of enemy becomes boring as it brings no new challenge. Similarly a friendly using a FSM could be easily manipulated into preforming a certain state if the player observes its conditions.

For more believable and less predictable transitions between states, *Fuzzy state machines* (FuSM) can be used, states are chosen on the highest value and values are assigned to states using fuzzy logic systems. Fuzzy logic is where rather than having statements to either be true or false, there is a range of variables between 0 and 1, where instead of either being sober or drunk you have degrees of sobriety. *Stack-based FSM's* can be used for interruption and returning to previous states. *Hierarchical FSMs* allow the grouping of states together of hierarchies making it easier to share transitions between states. Rather than having to connect a state such as "Eat" if the NPC is hungry, to every state it can be connected to a group states and called at any time from any state within that group.

2.4.2 Scripted Events

Scripted events are extremely popular in the games industry. RPGs such as *Neverwinter Nights* [50] use scripted events that will activate on a certain condition to give the NPCs a certain behaviour as part of a quest event or cinematic. Maps generally have triggers or way-points that when the player reaches activates a scripted event like some NPC running to you or some creature comes to fight. However, since these scripted events are predefined they do not come from the NPCs personality unless they are specifically tailored for that NPC which is the case in most RPGs, however, this is only for essential characters to the storyline or quest. For open world games these are used in only a select few places such as during a quest or when the player first enters a new city or encounters a special location.

2.4.3 AI Planners

Planners monitor the current state and plans the best action to perform next in order to reach its goal. The AI Planner is a technique to try capture the way a human would plan a strategy ahead. For example if a human lived 10 minutes walk from a shop he can either walk or drive, however, if the road is very busy he will decide to walk since it is the better choice of reaching his goal in the fastest way. This can be applied to AI, where they are in one state and they must reach a goal of perhaps killing a player but he is far away from the player and his available actions are to either melee or shoot, in order to shoot his gun must have ammo, if it does not he can reload his gun, if he does not have ammo to reload then he is unable to shoot and must melee, in order to melee, he must be close to the player which if he is not he must run up close to the player to achieve its goal.

Stanford Research Institute Problem Solver

STRIPS [51] is a planning algorithm that will search through possible world states by applying actions. The search is typically done backwards from the goal state because it is faster and more flexible with complex goals, a simple heuristic with A* can be used to make the planning process efficient. STRIPS was developed in Stanford in 1971 and became popular when Jeff Orkin used a slightly modified version called GOAP.

Goal-Oriented Action Planning

GOAP [52] is a variation of STRIPS created by Jeff Orkin for *F.E.A.R* [53]. It is a decision making algorithm that allows the NPC to not only decide what to do but how to do it. The NPC is given a goal and an action set it can use, and thus comes up with its own way of solving the goal. This ultimately make the NPCs actions less predictable and thus more believable. GOAP is a modified version of STRIPS, a cost per action is added.

A masters thesis student Long [54] states it is the first work that has compared GOAP and with another AI system, he compares it to FSMs and showed the usefulness and superiority of GOAP to FSMs.

2.4.4 Behaviour Tree's

Behaviour tree's (BT) have gained increased interest in commercial games in recent times. Games such as *Black & White* [42], *Halo* [55] and *Saints Row* [56] use BTs for their NPCs. An online journal on AI Planning [57]provides a detailed explanation of BTs. BTs are trees that represent a tasks breakdown into smaller tasks. The leaves of the tree is the interface between the AI logic and the game engine and include conditions and actions. The parent chooses which child tasks can be run and that childs action will be selected if the condition is true. They were developed due to problems with FSMs scaling up and not being able to reuse behaviours. BTs tasks have the advantage of being reusable and more efficient during run-time, some drawbacks are they can take longer to design than an action set for GOAP. Figure 2.3 shows an example of a behaviour tree for crossing a river.

Emotional Behaviour Trees

Johansson [58] presents an extension to behaviour trees technique to model gaming scenarios using emotion called the *Emotional Behaviour Tree* (EmoBT). The hope that this model can be used by game designers to bring emotion into their NPCs with the rise in popularity of behaviour trees, trying to incorporate emotions into them to make a more believable NPC would be desired. Three important factors were picked *Time discounting, risk perception* and *planning* since these are influenced by emotions. Using these factors Johansson creates an emotional selector which alters the priorities of the child nodes of the selector at run time. The results show that the selector takes

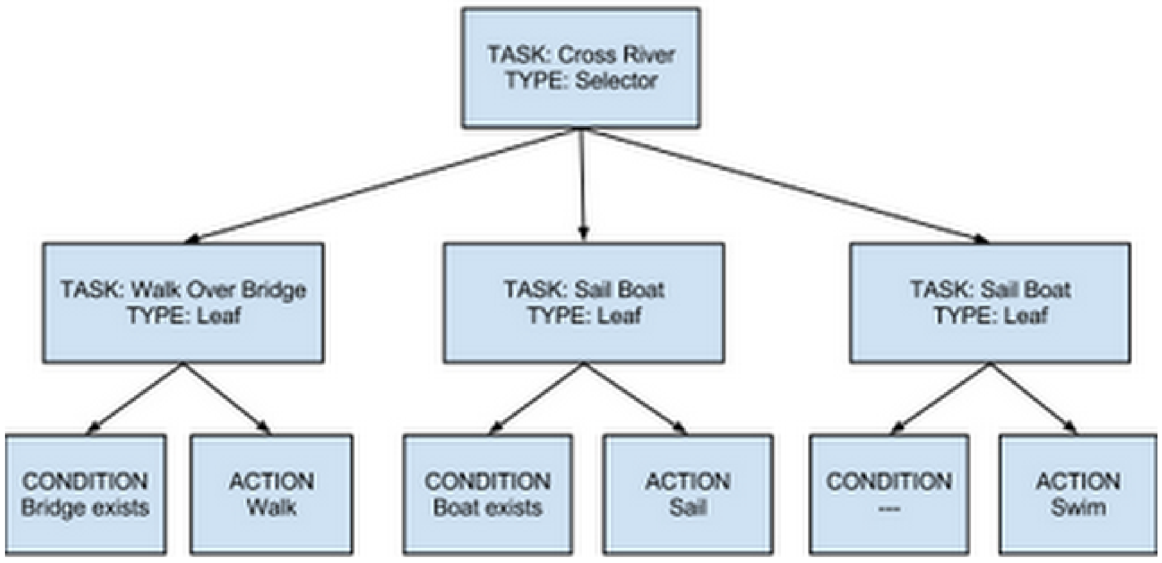


Figure 2.3: Behaviour Tree example for crossing a river.

the character's emotional state into consideration for the decision making. They show that the model which was fully implemented is efficient and scales O(n) with the size of the behaviour tree.

2.5 AI In Games

First a mention of the companions from *Half-Life 2* [14], *Bioshock Infinite* [15] and *The Last Of Us* [59]. These three games have the most advanced AI in games to date. They are currently rated as the most believable characters in games, however, the time to develop these agents is substantial and infeasible in many games other than first person shooters where the number of NPCs is limited. This is followed by a list of games that have innovated the area of AI for games.

2.5.1 Companions

A recent trend in games has the player who performs all the actions is guided by a companion. These companions are rather advanced and are essential for immersion of the player. They are the main way through which the story is told, and are given emotions which is visible for the player to try encapsulate him into the story. They provide a means for storytelling or informing the player of important details related to the story as the player progresses through the game. While many other games have a similar concept usually referred to as an "Escort mission" where the player must lead a AI from point A to B, there are usually many problems encountered, namely, the AI running to every enemy it sees far away and starting a fight, or only moves at a constant speed which frustrates the player as he is ultimately hindered by this AIs design.

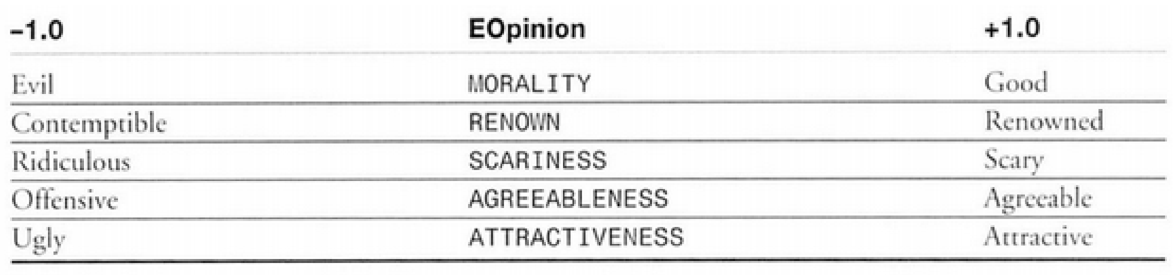
The most notable games which have this companion system are *Half-Lifes 2* - Alyx, *Bioshock Infinite* - Elizabeth and *The Last Of Us* - Ellie. In these games the player is rarely constricted by the companion, and they provide the player with an attachment to the game. For example Alyx in the second instalment of *Half-life 2 - Episode One* is primarily designed for cooperative play with the player. In a certain area the player is able to shine their flash light on an enemy and she will shot at that enemy. Elizabeths developers have said that she was inspired by Alyxs AI which at the time was unparalleled, some things that make her unique is while she does not help the player fight, she will sometimes find items which she will throw to the player to aid him. Her animations, design and voice acting are superb and it really gives the player the sense that he is escorting a teenage girl around a city she has never seen before with her interactions with the environment as she will look at something afar in a scene and run to it if she is interested and then run back to the player if he moves on. This gives the player an emotional attachment and gives the player a sense that the way she acts is very believable as a human. Similarly Ellies design, animation and voice acting are implemented perfectly to give the player a feeling that you are playing with a real character with real emotions.

2.5.2 Black & White

Black & White [42] made by Lionhead Studios had a primary focus around AI. The player plays as a God who controls over a civilization which is inhabited by villagers and you have the ability to create buildings and use miracles, however, the major part of this is that you have a creature which can learn from your actions. The main theme of the game is that you can either be good or evil and depending on your actions your creature will in turn be good or evil. The creature will perform actions which the creature observed you performing. The model Lionhead studios used was a *Belief-Desire-Intention* model (BDI) for the creatures learning and decision making. Where beliefs are attribute lists which store data about the world objects, desires are the goals to be fulfilled and opinions describe ways of satisfying a desire using a decision tree. The creature forms an intention by combining these three. For each desire the creature selects the belief that has the best opinion and forms an intention / goal. Opinions the creatures form are based on player feedback, the player is able to punish or reward the creature after it performs an action, for example if the creature eats a tree (it happens) if the player encourages the creature he will continue to do so.

2.5.3 Fable II

Fable II [43] also developed by Lionhead Studios, uses a similar system to Black & White, an opinion system. Like Black & White's creature, the player in Fable II also has a dog whose actions depend on the actions of the player, also like Black & White it uses the BDI system [60]. The NPCs throughout the world form opinions of the player through his actions. He can also flirt and give gifts to raise the opinion the villager has for the player. The player is able to alter his appearance by getting tattoos which contribute to his scariness value. The players physical appearance is also altered by his actions of good or evil which affects the way the villagers will think of the player. If he is good, he will be an angelic looking hero where the NPCs will cheer and run up to him to praise him. For evil characters the NPCs will flee in terror from him. This gives the player the ability to make his own presence felt in the world and that his actions will directly affect the NPCs around him. Fable uses a multidimensional opinion system where an overall opinion is formed from the following 5 values in figure 2.4.



-1.0	EOpinion	+1.0
Evil	MORALITY	Good
Contemptible	RENOWN	Renowned
Ridiculous	SCARINESS	Scary
Offensive	AGREEABLENESS	Agreeable
Ugly	ATTRACTIVENESS	Attractive

Figure 2.4: Fable Opinion System

2.5.4 F.E.A.R. : First Encounter Assault Recon

F.E.A.R [53] is a FPS developed in 2005 by Monolith Productions. F.E.A.R uses an A* algorithm for path-finding and a FSM but NPCs only have three states and yet are capable or a wide range of actions such as taking cover, blind firing from cover, using the environment to their advantage, appropriate use of grenades to get the player out of cover, the ability to communicate with each other for information and more. This is a lot of actions that a NPC can do in just three states shown in figure 2.5.

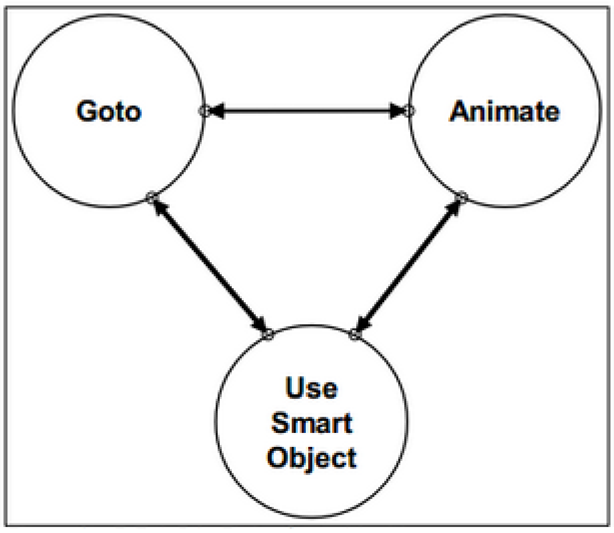
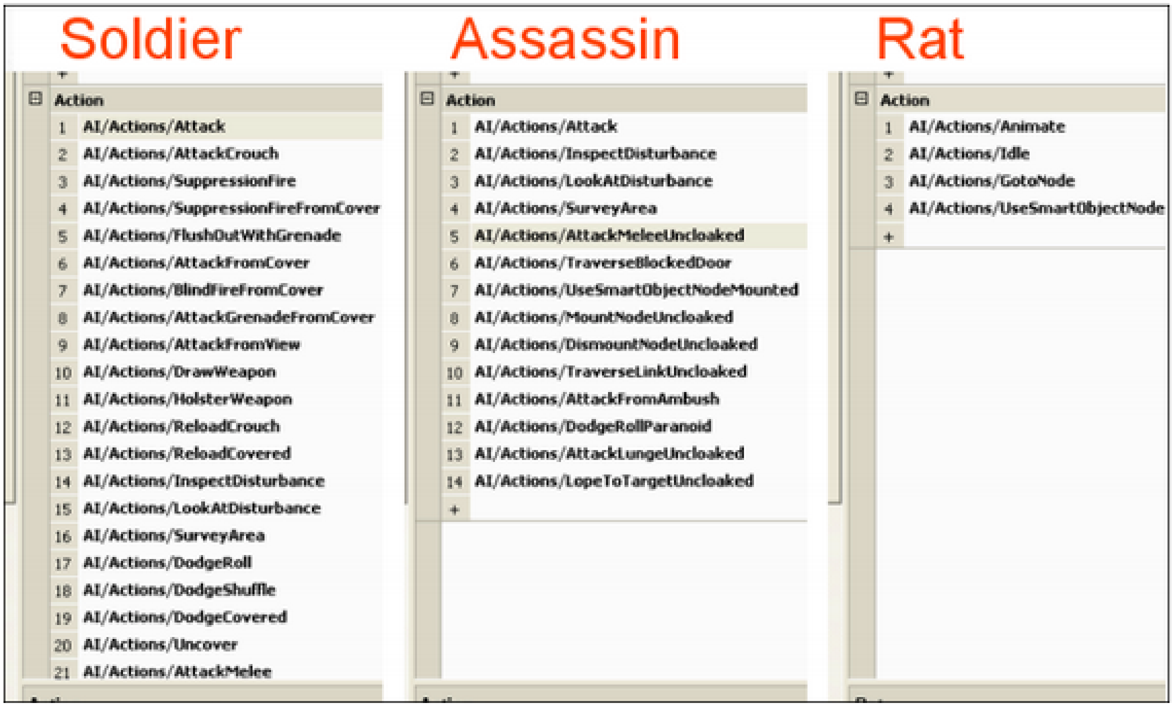


Figure 2.5: FSM in Fear

One state is "Use smart object" is for animations, leaving only two states of importance. F.E.A.R uses a planning system to let the agents know when it needs to transition states or what state to be in. While a FSM tells the agent exactly what to do in a given


Soldier

	Action
1	AI/Actions/Attack
2	AI/Actions/AttackCrouch
3	AI/Actions/SuppressionFire
4	AI/Actions/SuppressionFireFromCover
5	AI/Actions/FlushOutWithGrenade
6	AI/Actions/AttackFromCover
7	AI/Actions/BlindFireFromCover
8	AI/Actions/AttackGrenadeFromCover
9	AI/Actions/AttackFromView
10	AI/Actions/DrawWeapon
11	AI/Actions/HolsterWeapon
12	AI/Actions/ReloadCrouch
13	AI/Actions/ReloadCovered
14	AI/Actions/InspectDisturbance
15	AI/Actions/LookAtDisturbance
16	AI/Actions/SurveyArea
17	AI/Actions/DodgeRoll
18	AI/Actions/DodgeShuffle
19	AI/Actions/DodgeCovered
20	AI/Actions/Uncover
21	AI/Actions/AttackMelee

Assassin

	Action
1	AI/Actions/Attack
2	AI/Actions/InspectDisturbance
3	AI/Actions/LookAtDisturbance
4	AI/Actions/SurveyArea
5	AI/Actions/AttackMeleeUncloaked
6	AI/Actions/TraverseBlockedDoor
7	AI/Actions/UseSmartObjectNodeMounted
8	AI/Actions/MountNodeUncloaked
9	AI/Actions/DismountNodeUncloaked
10	AI/Actions/TraverseLinkUncloaked
11	AI/Actions/AttackFromAmbush
12	AI/Actions/DodgeRollParanoid
13	AI/Actions/AttackLungeUncloaked
14	AI/Actions/LopeToTargetUncloaked
+	

Rat

	Action
1	AI/Actions/Animate
2	AI/Actions/Idle
3	AI/Actions/GotoNode
4	AI/Actions/UseSmartObjectNode
+	

Figure 2.6: Action sets in F.E.A.R

state, a planner gives the agent goals and what actions it can do to give the agent the ability to reach the goal in its own way. F.E.A.Rs planner is called GOAP (2.4.3) which was developed by Jeff Orkin. In F.E.A.Rs development team they had only one AI programmer but many different types of AI. Their philosophy was that if they could develop a smart AI that would use the surroundings of the environment it would reduce the workload for having to manually program the AI for various levels and special cases since all they had to do was put it in an environment and it would accomplish its goal by using its surroundings. Also if they could develop an intelligent AI, incorporating that into a group should be easier than having to completely redo the model for group work.

A simple FSM would consist of a state like attack, which could simply tell the agent to fire at the player until he is killed but unless the programming specifically tells the agent to move to cover or other special cases the agent would not do this. While a planner would give the agent a goal to kill the player and only provide the agents with actions such as cover, duck, flank, etc. The agent could use these in a random and unpredictable way to achieve its goal. Giving different agents a different list of actions would result in unique and separate agents which would achieve their goal in very different ways. Figure 2.6 shows an image of three action sets for a Soldier, Assassin and a Rat.

Having this ability to create agents by just giving them action sets, the F.E.A.R team was able to implement a new agent late in the development by combining some of the previous action sets to create a new type of enemy with minimal effort. The benefits of a planner system like this over a FSM in terms of believability can be seen as the agents will rarely select the same action and thus come up with their own unique way to solve their goal. This gives the agents the ability to think for themselves rather than being designed specificity.

2.5.5 Façade

Façade [61] is a small game which demonstrates the use of a drama engine. The idea is to create a game for storytelling, like a play but one that the player is part of. It tries to follow the Freytag's Pyramid [62] in figure 2.7 approach for drama. The setting is a small apartment, a player who can interact with some objects, and with two NPCs a man and wife. Through the players actions, gestures or natural language, the player can interact with the scene and cause a different version of the story. Written in *A behaviour Language*(ABL), Façade uses what they call beats, which are reactive behaviours that will move the story forward each beat contains 10 - 100 behaviours. Based on the players actions the drama engine will try select the most appropriate beat to play next that will follow an Aristotelian dramatic arc. Façade has 27 beats such as *PlayerArrives, PayerFollowsTripToKitchen, EndingNoRevelations.* While only set in a small area and only 2 NPCs, the game is still quite large (roughly 800MB) due to voice acting and the number of possibilities that could occur, this would have numerous problems if it was to be incorporated into an actual AAA game. It does show potential of the use of drama engines for future use.

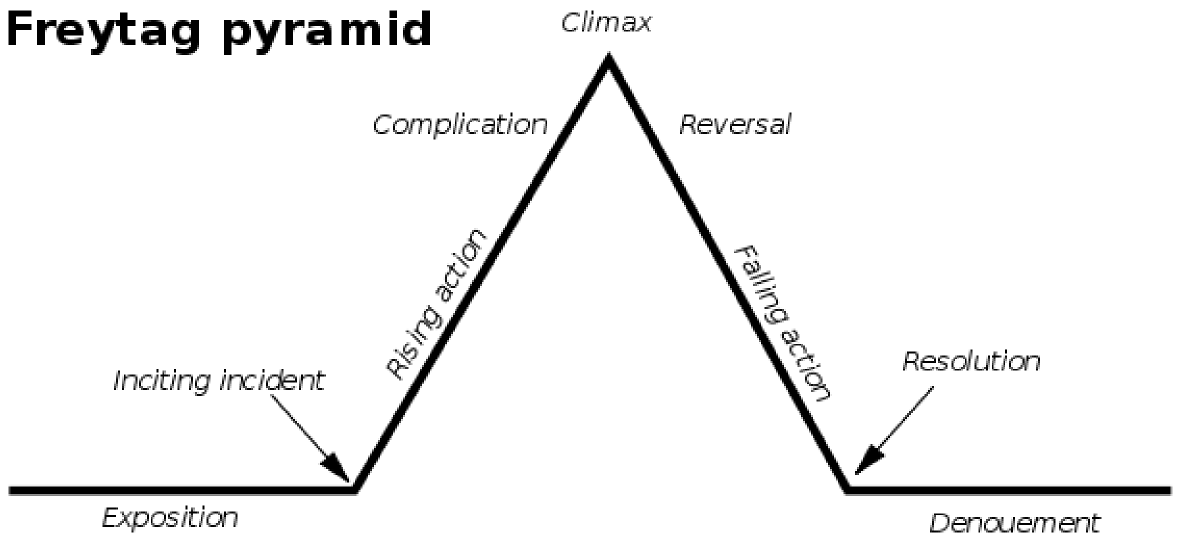


Figure 2.7: Freytags Pyramid showing the "'Five Act Structure"' of drama

2.6 Methods Used In Open World Games

Open world games AI is very different to First Person Shooters in that the area of the levels are much larger. This is mainly because in these games the main aspect is the game environment, and the number of characters that could be visible on the screen could be 30-40+ which could be intense with the use of a more realistic model, this is why a cheap efficient system for AI must be used. Primarily simple systems used are a FSM and some A* or way-point system for path-finding. Since there would possible be only a few variations such as friendlies and enemies the NPCs in most games would use the same FSM their actions and reactions would be identical. In recent years however, with improvement in AI technology developers are trying to move away from FSMs which have their own problems to use different techniques.

GOAP has also been implemented in open world games such as S.T.A.L.K.E.R [4] and Just Cause 2 [63]. Jeff Orkin, mentioned in his publications the benefit of planners for open world games [64].

2.6.1 S.T.A.L.K.E.R.

S.T.A.L.K.E.R. is a FPS Open World game, AI in Stalker is controlled by the *A-Life* system [65]. A-Life or Artificial Life is a study of living systems in an artificial environment. The world is populated with characters once the game is started each of these with their own life and routines. Like F.E.A.R, S.T.A.L.K.E.R uses a GOAP system for the NPCs to plan their next action. Because monsters rely extremely on animations which cannot be interrupted or blended, a Hierarchical FSM was used for them rather than the GOAP system. NPCs have a full life cycle, task accomplishment, combat, resting, feeding and sleeping. Monsters are created to migrate in large groups like a pack of wolfs. This gives the player a sense that he is in the middle of a real living environment.

2.6.2 Skyrim

Skyrim[3] uses a system called *Radiant AI* [66]. It is what they call a dynamic reaction to the players actions by both NPCs and the world in Skyrim. This has the ability to give the player some random quests, adjust an areas monster count depending on the skill of the player and ensures that the game is played differently every time. Part of it is that NPCs will often comment on previous achievements, if the player drops an item near a NPC they will ask "Why would you do that?". While this Radiant AI system is an improvement over previous systems in the Elder Scrolls series, Oblivion[67] and Morrowind [68] which mainly used scripts. The NPCs are still unaware of most of the players feats, for example once the player has beaten the games main quest line, or killed a dragon, some NPCs will threaten the player when they are clearly outmatched. For the NPCs living in villages they are giving *AI Packages* which they preform throughout the day. These AI packages are things like "Eat" or "WorkInTown" or "SleepInBed", the NPCs are giving a list of AI packages and a time in which they can preform them in. When evaluating these packages the system will check if it can perform the action now, if not it will evaluate the next if all packages fail the NPC will preform a default packages such as wander around on the spot. While this is an incredible system and allows the creation of NPCs relatively easy, since all the NPCs generally have the same packages they will preform the same actions in different locations. It also breaks the illusion of these NPCs when suddenly at 5pm the NPCs stop working to preform another action in unison. The impressive part of Skyrim's background AI is the realism they provide from afar. While there AI is rather simple, mostly consisting of move to an area and interact with the provided markers, to the player who observes them typically from afar or briefly they fulfil their role of giving the player the illusion of life in this world.

John Fallon, a previous I.E.T student last year wrote a dissertation on "Believable Behaviour of Background Characters in Open World Games" [69]. In his dissertation John Fallon outlines a model to increase the fidelity of these background NPCs in Skyrim. While he achieved this he noted some impediments which hindered his final project. John spent most of him time learning about the *Creation kit*, Skyrim's level editor and *Papyrus* which is Skyrim's scripting language and thus was unable to fully implement the model. This was mainly due to the lack of data structures in Papyrus. This dissertation is an extension of the work by John Fallon and attempt to create a working implementation of the model and to improve it.

2.7 Conclusion

Mentioned in this chapter are some models that are used in industry to create a believable character for the player to interact with. Some AAA titles as mentioned in the companions section (2.5.1) the work required for just these companions is immense, developers for BioShock Infinite had considered cutting Elizabeth because of the amount of work required and problems encountered through development. For an open world game developers can not spend this much time designing one personality, let alone a whole world populous and the resources for these NPCs must be minimal since the world for these open world games much bigger than the FPS levels. This is why for such large games the AI is very lacklustre and very predictable because most of the resources is spent on visually immersing the player in the environment.

While most research papers and in the industry focus on creating believable human-likeness, we cannot forget that in these virtual environments there are wildlife which if they do not appear believable can ruin the illusion. These little parts all add to the immersion of the game. If one piece is lacking it can have a negative effect on the others. In *Half-life* [70] throughout the world the player may come across cockroaches, while they appear minor and irrelevant, on closer inspection they have a rather intelligent AI. They react to the player and run away if he tries to approach, if they are in a dark room they will scatter when a light is turned on and will seek out for food such as dead enemies.

This dissertation attempts to create a model that can be applied to games such as Skyrim where actions selected by the agent are dependent on its current emotions. The idea is to improve the fidelity of these NPCs at a minimal cost to performance.

Chapter 3

Design

3.1 Introduction

This chapter presents the design of the Emotion Model created. The creation of the emotional model is the primary objective of this dissertation (see 1.3). First a overview of the whole model is presented then a breakdown of the major components that make up the model is discussed. Following this the core part of the model which is the *Action Selection Manager* is explained. A look at an example NPC produced from the model is then shown. Finally an architecture diagram is presented for easy visualization of the update loop for each NPC.

3.2 Defining the Model

As mentioned in section 2.6.2, this dissertation is a sort of extension of the work of John Fallon's work on an emotional model. John Fallon had designed an emotional model for use in Skyrim but had difficulty in implementing it due to constraints of Skyrim's scripting language, Papyrus. In the model he presented he had the concept of using a *state vector* that represents the NPCs mental and physical states, and *state modification vectors* that represent how an activity affects the mental and physical states of the NPC.

Similarly the emotional model created in this dissertation uses a similar concept of giving NPCs *Emotional States* such as Health, Mood, Energy and Satedness. These emotional states that the NPC will have are then altered when performing an *Action*, each actions contains *state modification values* that will change the NPC's emotional state values. The core part of the model is the *Action Selection Manager*. This is where an evaluation of the NPC's current emotional states take place and the next action is selected based on these emotional state values.

The major difference is the addition of *Traits* to the model, some example of traits are *Energetic*, *Gluttonous*, *Alcoholic* or *Social*. A full list can be seen in table 3.4. Traits are what will give the NPCs a unique personality. As Loyall [20] (see 2.1.1) mentioned about the characteristics of a believably NPC he stated that personality is the most important requirement, this is achieved through the use of traits. Just like humans we all have the same basic desires such as to sleep and eat, but our traits or habits are what define us, whether you prefer to play games or a physical sport comes down to the persons traits. It is the belief then that the NPCs should have the same emotional states and actions and that their whole personality is a result of these traits. These traits change the modification values so that two different NPCs performing the same action will potentially have different modification values depending on the traits of the NPCs.

3.3 Emotional States

Each NPC is given a number of emotional states such as the ones listed in the table 3.1. Each state has a 2D value from -1 to +1. As shown the mood state ranges between sad and happy. With the use of this when evaluating the NPC, the NPC could have a range of emotional feelings that represent how he is currently feeling, rather than just being tired, the NPC could be tired and hungry but happy or a different combination depending on the state values. This ultimately gives a more realistic approach to the NPCs and the next action he takes would take these factors into account. Using this method provides a more interesting NPC as it gives the NPC a huge range of possible feelings it can be experiencing at the same time.

State Name	**-1**	**+1**
Health	Injured	Healthy
Mood	Sad	Happy
Energy	Tired	Full
Satedness	Hungry	Full
Passivity	Aggressive	Passive
Sociability	Unsatisfied	Satisfied
Soberness	Drunk	Sober
Memory	Forgetful	Retentive

Table 3.1: Emotional States.

There are 8 emotional states that each NPC can have, the first four states *Health*, *Mood*, *Energy* and *Satedness* are the core emotional states. In the action selection manager these are the emotional states that are used when determining the next action. This is due to some actions such as drinking having negative effects on multiple of the secondary emotional states which would cause the NPC to never pick drinking as an action when evaluating the next action to pick. The secondary emotional states affect few actions so their use for determining the next action is not required. In special occasions they are used when evaluating the emotional states this is discussed in section 3.4.3.

These states can be easily added to. This will allow developers to be able to add or remove states to their desire.

3.4 Actions

Each NPC is able to perform a list of actions such as the ones listed above in table 3.2. Actions take up a big part of the model because these are what will define the NPCs behaviour. Each NPC will be assigned a job. This job is randomly selected from eight actions such as *Farmer, Blacksmith, Fisher* etc. This selected job is what the NPC will do for a living, however, if the NPC is a farmer, he also might enjoy fishing and will be able to do it as a past time, this is all dependent on the traits selected. Because of this, the NPCs will not be forced into sticking to several few actions, each NPC can preform any action. However, they will only pick them if they enjoy that action, again

Action Name	**Health**	**Mood**	**Energy**	**Satedness**	**Passivity**	**Sociability**	**Soberness**	**Memory**
Sleeping	+0.2	+ 0.1	+0.2	-0.2	+0.2	0.0	+0.3	+0.3
Eating	0.0	+0.1	0.0	+2.0	0.0	0.0	0.5	0.0
Drinking	0.5	2.0	0.3	0.5	-1.0	+2.0	-1.0	-1.0
Socializing	0.0	+1.0	-0.2	-0.2	0.0	+2.0	0.0	0.0
Idling	0.0	+0.1	+0.4	-0.1	0.0	0.0	0.0	0.0
Wandering	0.0	+0.1	+0.4	-0.1	0.0	0.0	0.0	0.0
Fighting	-1.0	0.0	-0.9	-0.2	+2.0	0.0	0.0	0.0
Fishing	0.0	0.8	-0.5	-0.2	0.0	0.0	0.0	0.0
Farming	0.0	0.8	-0.5	-0.2	0.0	0.0	0.0	0.0
Woodcutting	0.0	0.8	-0.5	-0.2	0.0	0.0	0.0	0.0
Blacksmithing	0.0	0.8	-0.5	-0.2	0.0	0.0	0.0	0.0
Mining	0.0	0.8	-0.5	-0.2	0.0	0.0	0.0	0.0
Hunting	0.0	0.8	-0.5	-0.2	0.0	0.0	0.0	0.0
Playing Music	0.0	0.8	-0.5	-0.2	0.0	0.0	0.0	0.0
Shop Keeping	0.0	0.8	-0.5	-0.2	0.0	0.0	0.0	0.0

Table 3.2: State Modification Values.

this is all based on the traits which will alter the way the NPC views that action in a negative or positive way. Actions are comprised of three parts, *Modification values*, *Activity Times* and *Multipliers.*

3.4.1 Modification Values

As mentioned when the NPC performs an action the emotional states are altered depending on the modification values corresponding to that state. These values are shown in table 3.2. They are changed based on the NPC's traits which are assigned at the start. The number of modification values depends on the number of emotional states that the NPC has. The modification values are applied once the action is selected in the action selection manager. The values for the jobs and past time actions are all the same and are only changed based on the traits that the NPC has. Idling and wandering are also the same and the NPC will choose one based on the trait *Curiosity.* These actions are what the NPC will mainly do to relax and gain some energy during the day time.

3.4.2 Activity Time

It is difficult to balance these modification values in a way that during during the night the NPC will be tired and will only sleep, because what if the NPC gets tired during the

Action Name	**11pm - 8am**	**8am - 5pm**	**5pm - 11pm**
Sleeping	4.0	0.0	0.0
Selected Job	0.0	4.0	0.0
Everything Else	1.0	1.0	1.0

Table 3.3: Action Activity Times.

day? Do we allow the NPC to sleep during the day? This can slightly ruin the realism if the NPCs start sleeping during the day or at irregular hours. For this reason activity time is used as a multiplier when evaluating the NPCs next action. Each action has an array for the whole day, mostly these are set to 1 so the NPC can perform them at any time. For sleeping and working, however, they are set to 4 during the night and day respectively and set to 0 otherwise. This encourages the NPC to perform these actions at certain times but not force him to which is important in order to have a NPC that will choose his own actions. Table 3.3 shows the activity times for the actions, the values are the multipliers stored in the activity times array.

3.4.3 Multipliers

Similar to Activity Times, multipliers are used as a way to encourage the NPC to perform a certain action without forcing the NPC to pick the action. The use of multipliers is only for certain actions such as eating and for relaxing actions such as idling or wandering. These multipliers gradually increase when the satedness or energy states are below a certain threshold so that without forcing the NPC to choose the action, it will likely become very high priority if not performed.

These can also be used to discourage the NPC from performing the tasks as well, so that certain actions can be less prioritised once they stop being beneficial. Eating, for instance, is generally very beneficial, but if the NPC is not hungry he will be discouraged from picking the action.

This could also be implemented for more interesting things such as if the NPC gets sick, his multiplier to eat could be lowered for the duration of the sickness. This would only require a change of the multiplier value rather than the action itself.

As mentioned in the emotional states section about the use of only 4 emotional states for choosing an action, the secondary emotional states can also be used depending on their multiplier. Generally they are set to 0 so they are not taken into account, but if the NPC is social from his traits, the social multiplier will be non zero and thus taken into account when calculating the delta. This will mean the NPC will prefer to do more social actions.

3.5 Traits

Traits are the defining characteristics of the NPCs. Each NPC is assigned randomly a list of traits from the overall traits list in table 3.4. These traits are what allow the NPCs to be procedurally generated which is an objective of this dissertation. Each trait affects the NPC in different ways, some traits will affect the NPC's activity times but most will affect the NPC's modification values, which results in each NPC choosing different actions for different reasons. Each trait can be true, false or not used. If true then the NPC has the positive of the trait such as for example *Energetic*, the NPC will be energetic giving him a +0.2 to all his energy modification values so that he will not get tired as fast. If false for energetic then the NPC will be lazy and -0.2 to all his energy modification values meaning that he will get tired more easily and will be likely to take more breaks and rest. If the trait is not used it has no effect on the NPC's values.

These traits are assigned randomly during initialization. This allows the NPCs to be procedurally generated in a controlled fashion. A random number between -1 and +1 is picked, if the value is between -1 and -0.4 the trait is false, if it is between +0.4 and +1 it is true, otherwise its not used. New traits could be easily added to this model and hooked into the model, allowing other developers to design their own NPCs. Traits can also be used as a talking point when talking to the player, the player could learn about the NPC's life by asking him about his hobbies or traits.

Trait Name	**Description**	**Effects**
Energetic	Whether the NPC is lazy or not	Energy State
Gluttonous	Whether the NPC gets more or less hungry performing actions	Satedness State
Alcoholic	Whether the NPC is less or more likely to drink	Mood & Soberness
Workaholic	Whether the NPC likes to work shorter or later	Activity Time
Social	Whether the NPC is dislikes or likes to perform social actions	Mood & Sociability
Insomniac	Whether the NPC sleeps during the day and works at night	Activity Time
Likes Work	Whether the NPC dislikes or likes his job	Mood
Curiosity	Whether the NPC prefers to wander or not	Idles or Wanders
Neuroticism	Whether the NPC has a tendency to get angry or not	Effects Passivity when working
Fisher	Whether the NPC dislikes or likes to fish	Mood
Farmer	Whether the NPC dislikes or likes to Farm	Mood
Etc.	*All Similar Pastime actions are determined by traits*	Mood

Table 3.4: Traits.

3.6 Action Selection Manager

The Action Selection Manager is the core part of the model. It is responsible for choosing the next action based on the NPC's current emotional state values. The main idea behind the creation of this manager is the idea that it should not be hard coded. Rather than setting threshold values for when the NPC should eat or sleep, it should be a decision the NPC makes himself. Figure 3.1 shows a flow chart of the action selection manager during the update loop.

During every update this action selection manager is called and evaluates the current emotional states. The first thing that the manager does is iterate through each action to find how much each action will change the NPCs overall emotional states.

This change or *Delta* is calculated for each state and added together for each action so that the total delta is acquired. The formula is shown in the next line.

(Clamp(*state value* + *state modification value*) - *state value)* * *state Multiplier*

For each state its modification value from the current action is added to it, then clamped to ensure it is within -1 and +1 this is subtracted from the current state value to obtain the total change that state will have, this is multiplied by a multiplier if there is one and then added with the other emotional state deltas.

This delta will be the overall change so it might be only be a small number such as +0.3. This delta value is then multiplied by the activity time which is generally 1, but for sleeping and working during their hours it is 4 this will give it a bigger value making it a more likely choice. This delta value is then stored in a list along with the action's ID. This *Delta List* stores the delta for each action, this list is then sorted in a descending order. This can be observed in figure 3.2.

3.6.1 Probability Model For Increased Believability

How the action selection manager selects the next action can be viewed in figure 3.2. For believability, rather than picking the first action from this list which would be the most beneficial action for the NPC, this can loose the believability of the NPC if he is always performing the best action for himself, it would make him more predictable. In the action selection manager a small random element is added. Using the top 4 items in the list, the deltas are added together for a *Total Delta*, this total delta is multiplied by a random number between 0 and 1, so that the action picked is a probability based on its delta. This is so that while the NPC will be performing something of benefit because only the top 4 items are picked, it is slightly random to add a little unpredictability. Since this is a probabilistic model generally the top item will be most likely picked if the activity time, or action multiplier are large enough. This way not only is there a slight randomness in what action is picked, but the NPC will also most likely perform a task such as eating if he is hungry.

A trait such as *Conscientiousness* [28] (from the Five Factor Model) could further expand this by altering the number of actions the random generator will pick from in the list making the NPC could be more spontaneous, choosing random actions at times, or organized choosing only the top action.

3.7 External Factors

Rather than having the energy only decrease whenever the NPC performs an action, an idea of having a constant decay throughout the idea was proposed. The idea that as the day progresses the NPC's energy would slowly decay and possibly decay faster at night to try encourage the NPC to sleep. This same concept could be applied to the NPC's sociability state, if he goes for extended periods without meeting another person his sociability could slowly decay, causing him to get lonely so that he would be more encouraged to preform a social activity to remedy this.

This could also be hooked up into a weather system, where depending on the current weather it could effect the NPC's current mood, e.g. rain would upset the NPC if he is caught outside getting wet.

3.8 Created NPCs

Due to the way this model is set up, the NPCs can pick any action they want. Their perception of what actions they like are determined by traits during initialization. This is a much more ideal system than telling the NPC he can only farm. It yields interesting NPCs: e.g. you have a NPCs whose job is to farm. His traits determine that he does not like farming and every time he farms it will make him unhappy. When he is unhappy he will tend to pick an action that will make him happy which could be drink, but if he does not like drinking he might decide to do another pastime such as hunt or fish whether he likes it or not. There is always the possibility that he might pick something that he does not enjoy but this would be rare because the actions he would not like during evaluation would be at the bottom of the list.

Due to the number of traits that NPCs can have, it is hard to define them such as "The Drunken Farmer" which was a NPC that John Fallon had theorized. The farmer would hate farming so every time he farmed it made him unhappy and drove him to drinking. A very interesting and unique character. A similar NPC, an unhappy blacksmith has been created randomly which is discussed later in section 5.1.

With the number of traits currently in the model by having them randomized at the start it is unlikely that such a character will be created. However, the NPCs created with this model are much more complex and interesting because when they preform actions they might seem odd to the observer without knowledge of what the NPC likes or dislikes. This creates a more believable NPC rather than a stereotypical one.

3.9 Procedural Content Generation

As mentioned a goal was to support procedural generation of the NPCs (see 1.3.2). This will allow the model to be hooked up to a game engine and the NPCs will be automatically generated. This reduces the effort required by developers to create the NPCs allowing their focus onto other tasks. Another key factor in the use of *Procedural Content Generation* (PCG) is the re-playability of the game for the player. This ultimately avoids any two game plays to have identical NPCs giving the game more variability. The developer would only need to add custom traits they wish the NPCs to be able to have and during run time the NPC will pick these traits randomly altering their personality. This would only occur at the very start of a game, for example like Skyrim once the game is started, all the NPCs would be assigned their traits which would be saved so the NPCs keep their personality throughout that save. If the player decides to restart a new save, the NPCs would once again be recreated and assigned random personalities, making the play through slightly different to the last.

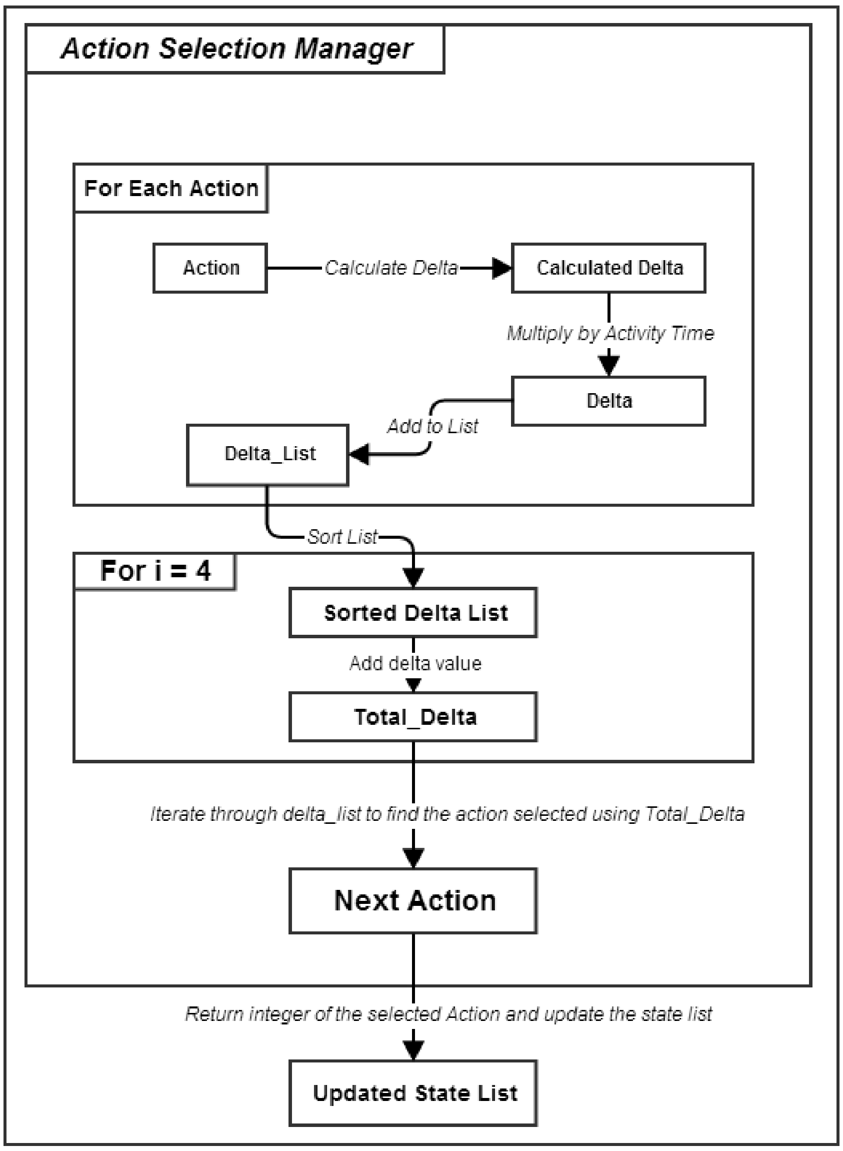


Figure 3.1: Action Selection Manager.

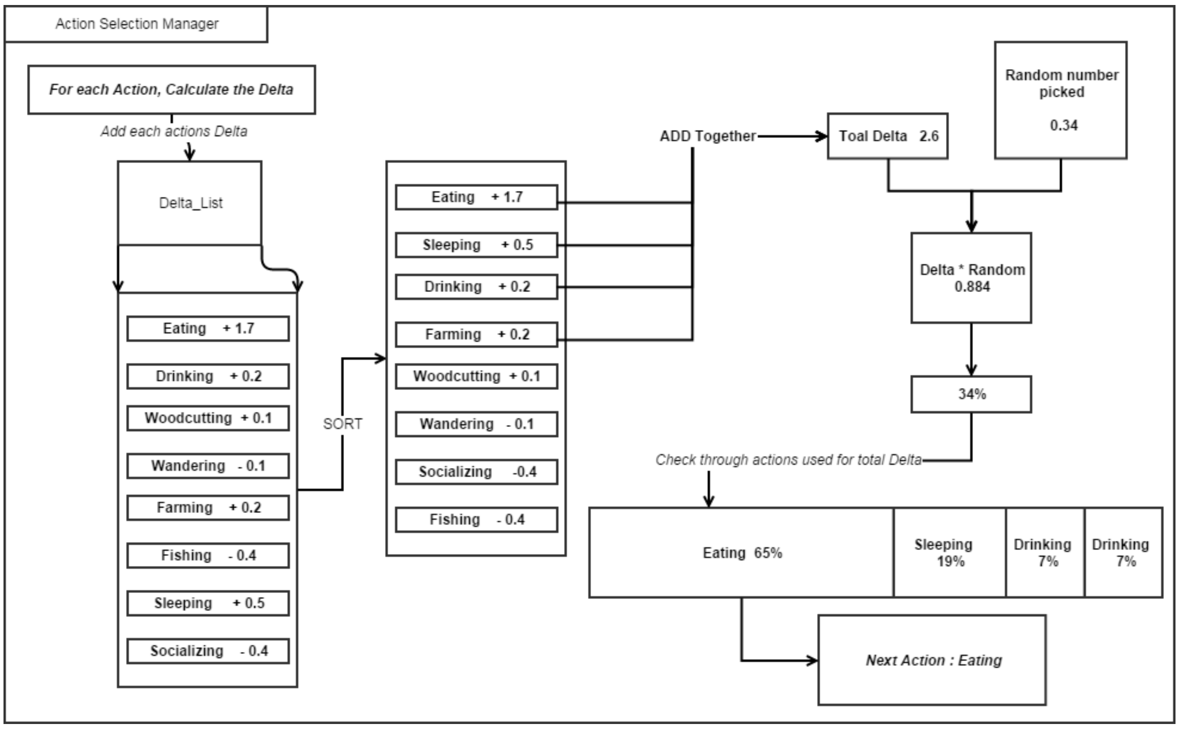


Figure 3.2: Diagram demonstrating how the Delta values are used for picking the next action.

Chapter 4

Implementation

This chapter is concerned with implementation of the Emotional Model. First a mention on the choice of platform used for implementing the model. Details are mentioned on the design choices taken during implementation and some problems encountered. Following this are some extensions made with the model such as exporting to a game environment and the ability to create custom NPCs during run time.

4.1 Platform Selection

This dissertation was originally intended to implement the emotion model in Skyrim. Skyrim was chosen because of the infrastructure the creation kit has and the relative ease of its use. While John Fallon had encountered problems using Papyrus, Skyrim's scripting language, mainly due to its lack of proper data structures he noted that a extension for the creation kit called *Skyrim Script Extender* (SKE) [71] would improve this. The SKE is intended to allow modders of Skyrim to integrate their code with the Skyrim engine allowing much more control to the developer, it also allows the developer to write code in C++ allowing full control of C++ data structures. At first this seemed the ideal solution, however on further investigation and trial it was deemed to be to complex and awkward to use. While Papyrus is fully documented, the SKE has little documentation and trying to compile a simple program required too much work.

Initial progress was slow due to trying to understand and learn how to use the creation kit. After much investigation, it was decided that a move to another platform was the best choice. Tiarnán McNulty, another student on a similar project [72], suggested a move to Unity3D [73]. Within a week a village had been created by Tiarnán which could be used for testing. Unity uses C#, and since the move to Unity a very quick prototype model was created.

4.2 Implementation Of The Model

Since this model is mostly concerned about increasing the believability of NPC's selected actions, a world environment was not necessary for initial testing. A simple flat plane and some NPCs in the form of boxes was all that was required (figure 4.1). In the next sections the design choices are outlined, as well as how the program was debugged.

4.2.1 Design Choices

Some design choices were mostly in the *Action Selection Manager.* The change from using all eight states to only using the core four when evaluating the delta for each action was primarily due to the NPC not wanting to ever drink due to the negative effects it has, while this is true even in real life, these negative effects are often over looked and it is still performed.

Another design choice was the addition of the random element in the Action Selection Manager, when trying to create believable agents, its not very believable if the agent knows exactly what the best action to take is. With this in mind the random element was added so that while he will still choose a action that is high up on the list so it will be beneficial, it will not always be the most beneficial action. With the addition of this, it is believed that a more believable and less predictable NPC is created.

With the addition of this random element in the action selection manager and the use of multipliers, to avoid simple multiplication errors that could occur with negative numbers, the range that the emotional state values use from -1 to +1, was changed to

0 to +2. This was just to avoid any potential problems that might be overlooked or go unnoticed.

Another major choice was implementing the model so that it was very easy to apply to other projects and games. With this in mind an excel file was created that will contain the emotional states, their initial values and the list of actions with the initial modification values (see 4.3.4). The idea behind this is that a developer would only have to change this excel sheet to add or remove emotion states or actions that the NPC has without having to code.

While the goal was to procedurally generate the NPCs, this made the model a little less generic. During the final stages of the dissertation a prototype "Customizable NPC Creator" was created. This is a slightly modified script which allows the developer to create and test custom NPCs during run time, and simulate their days to evaluate the traits selected.

4.2.2 Debugging

The main part of this dissertation was debugging, figuring out why the NPC decided to perform a certain action at certain times, then tweaking the values in an attempt to improve. One minor problem with Unity is the inability to debug during run time with the use of breakpoints. During the initial periods debugging the model was done during run time, visually representing the emotional states beside each cube as shown in figure 4.1.

Later the NPC's day was simulated during initialization and the actions selected are printed to a text document. Expanding on this all values about the NPC such as emotional state values, selected actions and information in the Action Selection Manager such as the deltas for each action and the delta list were printed to the text document if required. With this information it was possible to figure out why the NPC decided certain actions at certain times and the problem could be fixed. In figure 4.2 and figure 4.3 these images of the text documents are printed out for each NPC. Figure 4.2 lists the NPC's picked traits, the selected job, the state modification values after traits are applied to them and the activity times for each action. Shown in figure 4.3 is a

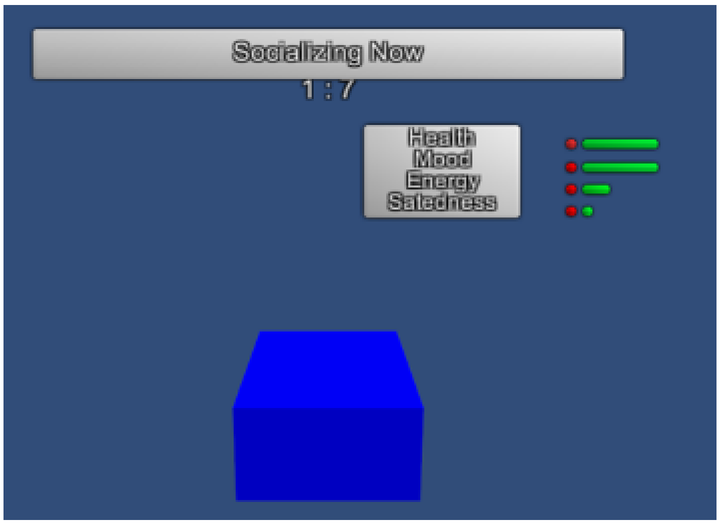


Figure 4.1: Initial Debugging of the Program.

less detailed version that just shows the selected actions and the current state values that the NPC chose every day. Figure 4.4 is a snippet of 3 actions the NPC choose, under these actions are all the actions added to the delta list, with their delta value in brackets. Beside the modification values for each action is the delta, which shows the change that action will have on the emotional states by followed by the multiplier in brackets.

Before the program starts a simulated day is printed out, then every action the NPC takes after that is also printed out.

4.3 Extensions of the Model

The following section is concerned with details about extensions of the emotional model taken in this dissertation.

4.3.1 Integration Of The Model Into A Game Environment

Since this model is for a game environment, during the final stages of this dissertation, it was implemented into a Unity project Tiarnán McNulty had created for his memory model. The Emotion Model was a single script that could be easily added to NPCs. Once the actions in the memory model were implemented into the world, all that was required was to get the implemented goal scheduler to accept values from the emotion model. The NPCs in the world use a goal orientated planner for decision making (see 2.4.3). During initial stages of development, it was discussed about the merging of the two projects towards to end as a demonstration. With this in mind a memory state was added that could be used to effect the memory model created. Sadly, due to time constraints, this was not fully implemented, but all the pieces are in place for it. Figure 4.5 shows an image of one of the NPCs passing by the player. Information about the NPC is visible such as what action he is currently doing and what his current emotional state values are.

Another feature also in place is the ability for the NPCs to visually represent how they feel. The NPC models used in the world have several different facial meshes which can be used to express how the NPC is currently feeling. Hooking this up to the emotion model was trivial and demonstrates how it could be used to alter what animations the NPC could use.

4.3.2 Custom Creation Of NPCs

All the NPCs before this have been randomly generated with random traits. However, what if the developer wanted to create their own NPCs and choose their own traits? With this in mind the idea of being able to create NPCs during run time and control what traits they have came to mind. While this may not be useful during a game it does help to test the Emotional Model and the selected traits with the ability to simulate days. With direct control over what traits are applied, the NPC being created can simulate a specified number of days to test if the traits have a certain effect. Once the NPC is to the liking of the developer, it can be created in the environment. Figure 4.6 shows the dialogue box that is used when creating an NPC.

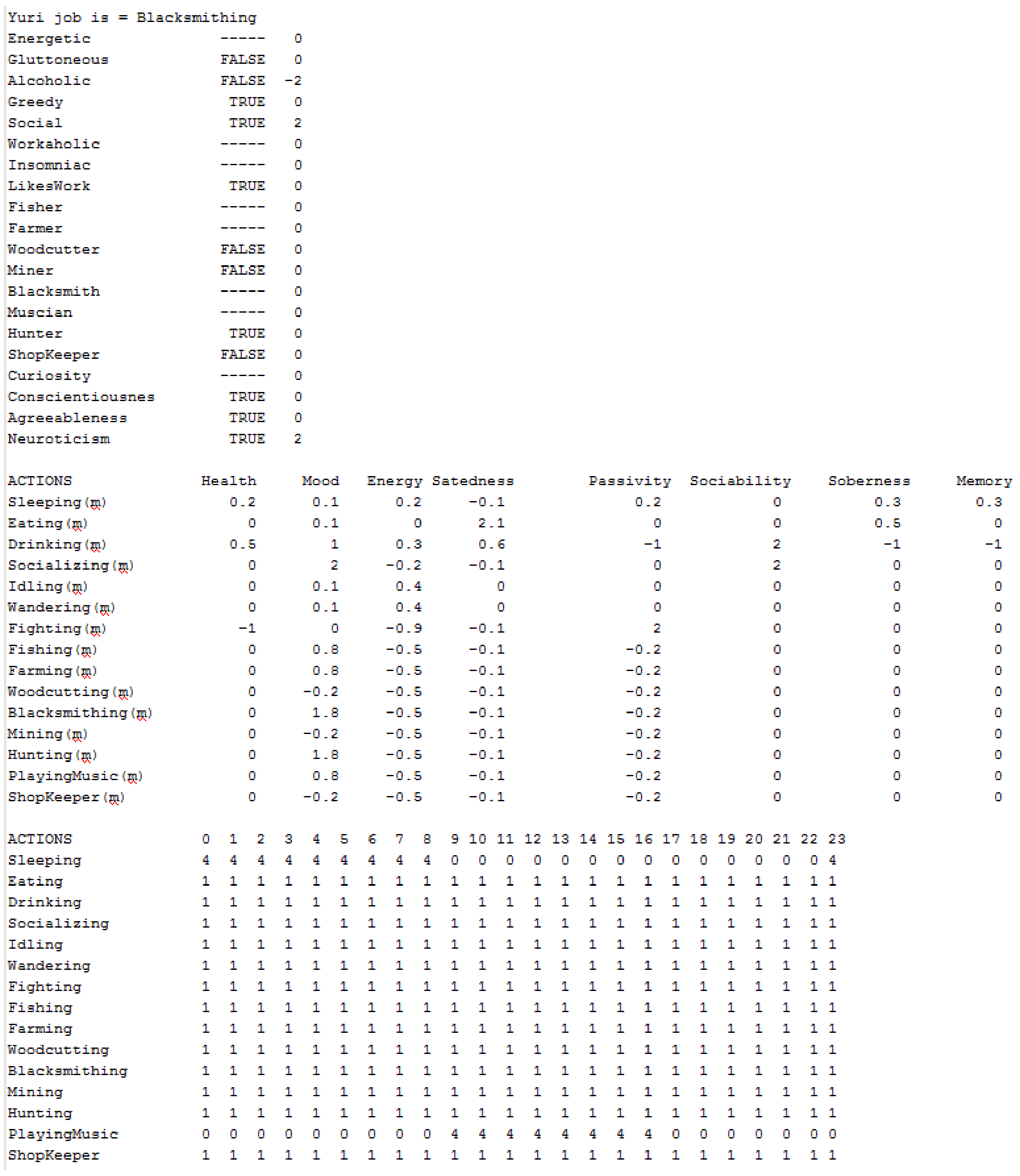
4.3.3 NPC Manager

The creation of an NPC manager class, which would take care of updating all the NPC's was done primarily for efficiency. Due to the NPCs only being updated once every hour, rather than calling update for every NPC on the hour, this allows the update calls to be staggered to avoid a potential bottleneck or stagger of game play. This NPC manager can also be used during initialization of the NPCs to equally spread out the jobs the NPCs select and avoid a potentially having all the NPC's randomly pick musician as a job and the village would starve as they have no fishers etc.

4.3.4 Excel Sheet For Initial Values

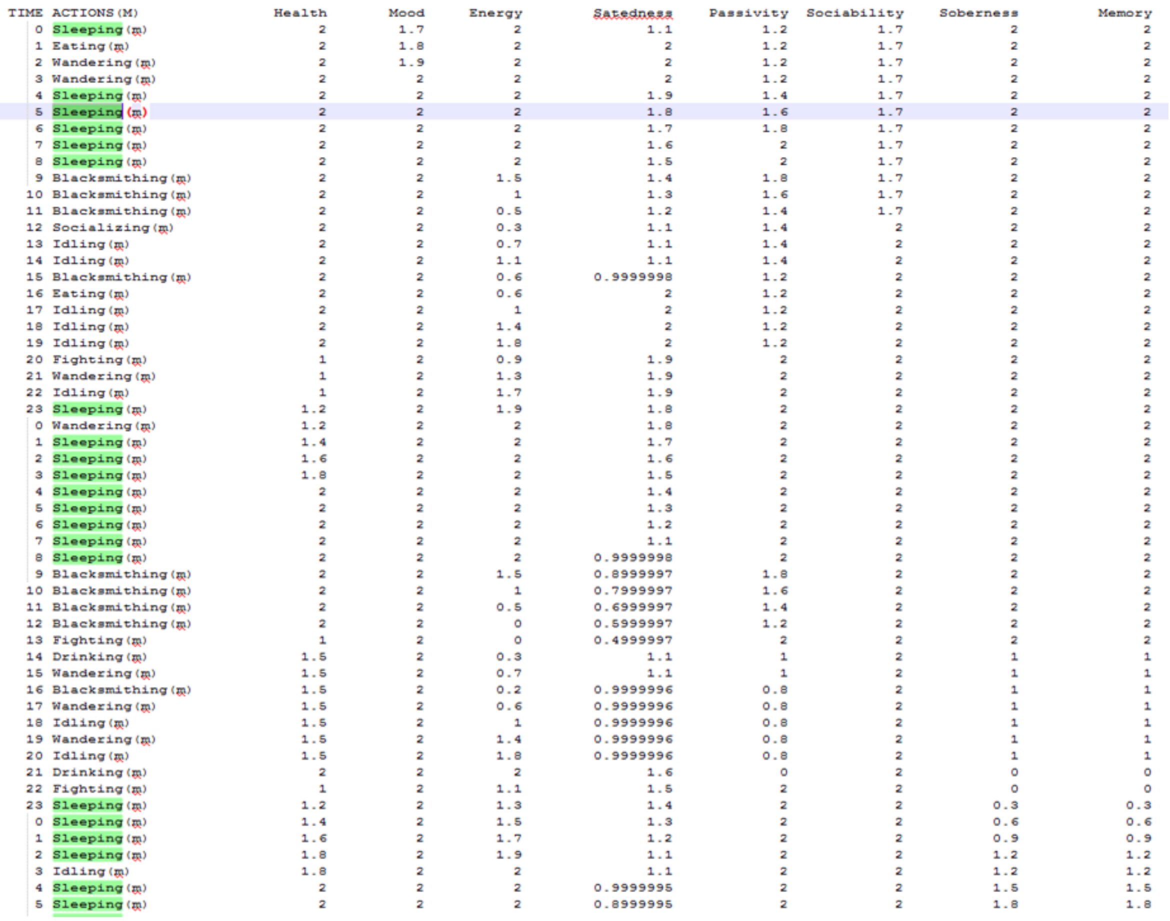
As mentioned in section 1.3 in the creating of this emotional model an emphasis was placed on making it generic so that developers could use it easily. To avoid having the emotional states and actions hard coded, the model was rewritten to read in a Comma-separated values (CSV) file. This CSV file contains all the information about the emotional states and actions the NPCs can have. An excel sheet can be used to add or remove emotional states and actions as well as modify their values. This excel sheet can easily be exported to a CSV file which is read in during the models initialization. The model then stores the emotional states and action values.

This allows the developers to only have to change this one excel sheet if they wish to add another action or emotional state. Figure 4.7 is the excel sheet used throughout this dissertation.



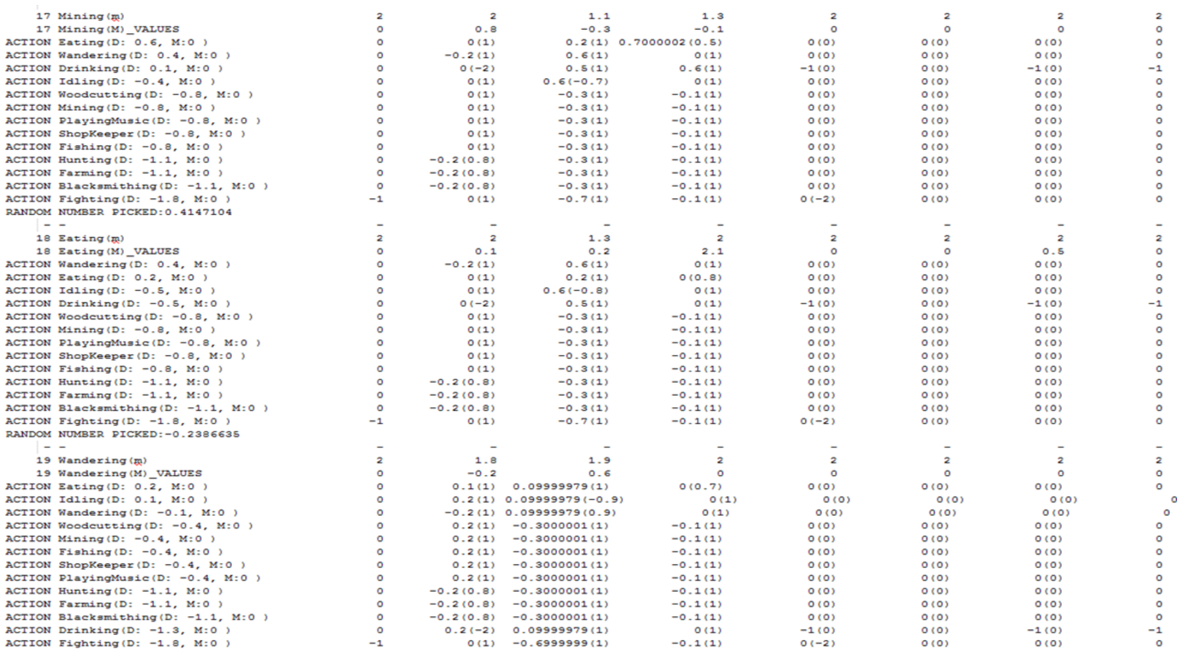
```
Yuri job is = Blacksmithing
Energetic              -----   0
Gluttoneous            FALSE   0
Alcoholic              FALSE  -2
Greedy                  TRUE   0
Social                  TRUE   2
Workaholic             -----   0
Insomniac              -----   0
LikesWork               TRUE   0
Fisher                 -----   0
Farmer                 -----   0
Woodcutter             FALSE   0
Miner                  FALSE   0
Blacksmith             -----   0
Muscian                -----   0
Hunter                  TRUE   0
ShopKeeper             FALSE   0
Curiosity              -----   0
Conscientiousnes        TRUE   0
Agreeableness           TRUE   0
Neuroticism             TRUE   2

ACTIONS                 Health     Mood   Energy Satedness       Passivity  Sociability     Soberness     Memory
Sleeping(m)                0.2      0.1      0.2      -0.1             0.2            0           0.3        0.3
Eating(m)                    0      0.1        0       2.1               0            0           0.5          0
Drinking(m)                0.5        1      0.3       0.6              -1            2            -1         -1
Socializing(m)               0        2     -0.2      -0.1               0            2             0          0
Idling(m)                    0      0.1      0.4         0               0            0             0          0
Wandering(m)                 0      0.1      0.4         0               0            0             0          0
Fighting(m)                 -1        0     -0.9      -0.1               2            0             0          0
Fishing(m)                   0      0.8     -0.5      -0.1            -0.2            0             0          0
Farming(m)                   0      0.8     -0.5      -0.1            -0.2            0             0          0
Woodcutting(m)               0     -0.2     -0.5      -0.1            -0.2            0             0          0
Blacksmithing(m)             0      1.8     -0.5      -0.1            -0.2            0             0          0
Mining(m)                    0     -0.2     -0.5      -0.1            -0.2            0             0          0
Hunting(m)                   0      1.8     -0.5      -0.1            -0.2            0             0          0
PlayingMusic(m)              0      0.8     -0.5      -0.1            -0.2            0             0          0
ShopKeeper(m)                0     -0.2     -0.5      -0.1            -0.2            0             0          0

ACTIONS                 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Sleeping                4  4  4  4  4  4  4  4  4  0  0  0  0  0  0  0  0  0  0  0  0  0  0 4
Eating                  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1 1
Drinking                1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1 1
Socializing             1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1 1
Idling                  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1 1
Wandering               1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1 1
Fighting                1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1 1
Fishing                 1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1 1
Farming                 1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1 1
Woodcutting             1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1 1
Blacksmithing           1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1 1
Mining                  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1 1
Hunting                 1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1 1
PlayingMusic            0  0  0  0  0  0  0  0  0  4  4  4  4  4  4  4  0  0  0  0  0  0  0 0
ShopKeeper              1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1 1
```

Figure 4.2: Top half of text document showing the NPCs characteristics.



TIME	ACTIONS (M)	Health	Mood	Energy	Satedness	Passivity	Sociability	Soberness	Memory
0	Sleeping (m)	2	1.7	2	1.1	1.2	1.7	2	2
1	Eating (m)	2	1.8	2	2	1.2	1.7	2	2
2	Wandering (m)	2	1.9	2	2	1.2	1.7	2	2
3	Wandering (m)	2	2	2	2	1.2	1.7	2	2
4	Sleeping (m)	2	2	2	1.9	1.4	1.7	2	2
5	Sleeping (m)	2	2	2	1.8	1.6	1.7	2	2
6	Sleeping (m)	2	2	2	1.7	1.8	1.7	2	2
7	Sleeping (m)	2	2	2	1.6	2	1.7	2	2
8	Sleeping (m)	2	2	2	1.5	2	1.7	2	2
9	Blacksmithing (m)	2	2	1.5	1.4	1.8	1.7	2	2
10	Blacksmithing (m)	2	2	1	1.3	1.6	1.7	2	2
11	Blacksmithing (m)	2	2	0.5	1.2	1.4	1.7	2	2
12	Socializing (m)	2	2	0.3	1.1	1.4	2	2	2
13	Idling (m)	2	2	0.7	1.1	1.4	2	2	2
14	Idling (m)	2	2	1.1	1.1	1.4	2	2	2
15	Blacksmithing (m)	2	2	0.6	0.9999998	1.2	2	2	2
16	Eating (m)	2	2	0.6	2	1.2	2	2	2
17	Idling (m)	2	2	1	2	1.2	2	2	2
18	Idling (m)	2	2	1.4	2	1.2	2	2	2
19	Idling (m)	2	2	1.8	2	1.2	2	2	2
20	Fighting (m)	1	2	0.9	1.9	2	2	2	2
21	Wandering (m)	1	2	1.3	1.9	2	2	2	2
22	Idling (m)	1	2	1.7	1.9	2	2	2	2
23	Sleeping (m)	1.2	2	1.9	1.8	2	2	2	2
0	Wandering (m)	1.2	2	2	1.8	2	2	2	2
1	Sleeping (m)	1.4	2	2	1.7	2	2	2	2
2	Sleeping (m)	1.6	2	2	1.6	2	2	2	2
3	Sleeping (m)	1.8	2	2	1.5	2	2	2	2
4	Sleeping (m)	2	2	2	1.4	2	2	2	2
5	Sleeping (m)	2	2	2	1.3	2	2	2	2
6	Sleeping (m)	2	2	2	1.2	2	2	2	2
7	Sleeping (m)	2	2	2	1.1	2	2	2	2
8	Sleeping (m)	2	2	2	0.9999998	2	2	2	2
9	Blacksmithing (m)	2	2	1.5	0.8999997	1.8	2	2	2
10	Blacksmithing (m)	2	2	1	0.7999997	1.6	2	2	2
11	Blacksmithing (m)	2	2	0.5	0.6999997	1.4	2	2	2
12	Blacksmithing (m)	2	2	0	0.5999997	1.2	2	2	2
13	Fighting (m)	1	2	0	0.4999997	2	2	2	2
14	Drinking (m)	1.5	2	0.3	1.1	1	2	1	1
15	Wandering (m)	1.5	2	0.7	1.1	1	2	1	1
16	Blacksmithing (m)	1.5	2	0.2	0.9999996	0.8	2	1	1
17	Wandering (m)	1.5	2	0.6	0.9999996	0.8	2	1	1
18	Idling (m)	1.5	2	1	0.9999996	0.8	2	1	1
19	Wandering (m)	1.5	2	1.4	0.9999996	0.8	2	1	1
20	Idling (m)	1.5	2	1.8	0.9999996	0.8	2	1	1
21	Drinking (m)	2	2	2	1.6	0	2	0	0
22	Fighting (m)	1	2	1.1	1.5	2	2	0	0
23	Sleeping (m)	1.2	2	1.3	1.4	2	2	0.3	0.3
0	Sleeping (m)	1.4	2	1.5	1.3	2	2	0.6	0.6
1	Sleeping (m)	1.6	2	1.7	1.2	2	2	0.9	0.9
2	Sleeping (m)	1.8	2	1.9	1.1	2	2	1.2	1.2
3	Idling (m)	1.8	2	2	1.1	2	2	1.2	1.2
4	Sleeping (m)	2	2	2	0.9999995	2	2	1.5	1.5
5	Sleeping (m)	2	2	2	0.8999995	2	2	1.8	1.8

Figure 4.3: Bottom half of the text document showing the selected actions the NPC chose.



```
    17 Mining(m)                                 2               2             1.1             1.3               2               2               2       2
    17 Mining(M)_VALUES                          0             0.8            -0.3            -0.1               0               0               0       0
ACTION Eating(D: 0.6, M:0 )                      0            0(1)          0.2(1) 0.7000002(0.5)           0(0)            0(0)            0(0)       0
ACTION Wandering(D: 0.4, M:0 )                   0         -0.2(1)          0.6(1)            0(1)            0(0)            0(0)            0(0)       0
ACTION Drinking(D: 0.1, M:0 )                    0           0(-2)          0.5(1)          0.6(1)           -1(0)            0(0)           -1(0)      -1
ACTION Idling(D: -0.4, M:0 )                     0            0(1)       0.6(-0.7)            0(1)            0(0)            0(0)            0(0)       0
ACTION Woodcutting(D: -0.8, M:0 )                0            0(1)         -0.3(1)         -0.1(1)            0(0)            0(0)            0(0)       0
ACTION Mining(D: -0.8, M:0 )                     0            0(1)         -0.3(1)         -0.1(1)            0(0)            0(0)            0(0)       0
ACTION PlayingMusic(D: -0.8, M:0 )               0            0(1)         -0.3(1)         -0.1(1)            0(0)            0(0)            0(0)       0
ACTION ShopKeeper(D: -0.8, M:0 )                 0            0(1)         -0.3(1)         -0.1(1)            0(0)            0(0)            0(0)       0
ACTION Fishing(D: -0.8, M:0 )                    0            0(1)         -0.3(1)         -0.1(1)            0(0)            0(0)            0(0)       0
ACTION Hunting(D: -1.1, M:0 )                    0       -0.2(0.8)         -0.3(1)         -0.1(1)            0(0)            0(0)            0(0)       0
ACTION Farming(D: -1.1, M:0 )                    0       -0.2(0.8)         -0.3(1)         -0.1(1)            0(0)            0(0)            0(0)       0
ACTION Blacksmithing(D: -1.1, M:0 )              0       -0.2(0.8)         -0.3(1)         -0.1(1)            0(0)            0(0)            0(0)       0
ACTION Fighting(D: -1.8, M:0 )                  -1            0(1)         -0.7(1)         -0.1(1)           0(-2)            0(0)            0(0)       0
RANDOM NUMBER PICKED:0.4147104
    - -                                          -               -               -               -               -               -               -       -
    18 Eating(m)                                 2               2             1.3               2               2               2               2       2
    18 Eating(M)_VALUES                          0             0.1             0.2             2.1               0               0             0.5       0
ACTION Wandering(D: 0.4, M:0 )                   0         -0.2(1)          0.6(1)            0(1)            0(0)            0(0)            0(0)       0
ACTION Eating(D: 0.2, M:0 )                      0            0(1)          0.2(1)          0(0.8)            0(0)            0(0)            0(0)       0
ACTION Idling(D: -0.5, M:0 )                     0            0(1)       0.6(-0.8)            0(1)            0(0)            0(0)            0(0)       0
ACTION Drinking(D: -0.5, M:0 )                   0           0(-2)          0.5(1)            0(1)           -1(0)            0(0)           -1(0)      -1
ACTION Woodcutting(D: -0.8, M:0 )                0            0(1)         -0.3(1)         -0.1(1)            0(0)            0(0)            0(0)       0
ACTION Mining(D: -0.8, M:0 )                     0            0(1)         -0.3(1)         -0.1(1)            0(0)            0(0)            0(0)       0
ACTION PlayingMusic(D: -0.8, M:0 )               0            0(1)         -0.3(1)         -0.1(1)            0(0)            0(0)            0(0)       0
ACTION ShopKeeper(D: -0.8, M:0 )                 0            0(1)         -0.3(1)         -0.1(1)            0(0)            0(0)            0(0)       0
ACTION Fishing(D: -0.8, M:0 )                    0            0(1)         -0.3(1)         -0.1(1)            0(0)            0(0)            0(0)       0
ACTION Hunting(D: -1.1, M:0 )                    0       -0.2(0.8)         -0.3(1)         -0.1(1)            0(0)            0(0)            0(0)       0
ACTION Farming(D: -1.1, M:0 )                    0       -0.2(0.8)         -0.3(1)         -0.1(1)            0(0)            0(0)            0(0)       0
ACTION Blacksmithing(D: -1.1, M:0 )              0       -0.2(0.8)         -0.3(1)         -0.1(1)            0(0)            0(0)            0(0)       0
ACTION Fighting(D: -1.8, M:0 )                  -1            0(1)         -0.7(1)         -0.1(1)           0(-2)            0(0)            0(0)       0
RANDOM NUMBER PICKED:-0.2386635
    - -                                          -               -               -               -               -               -               -       -
    19 Wandering(m)                              2             1.8             1.9               2               2               2               2       2
    19 Wandering(M)_VALUES                       0            -0.2             0.6               0               0               0               0       0
ACTION Eating(D: 0.2, M:0 )                      0          0.1(1)    0.09999979(1)         0(0.7)            0(0)            0(0)            0(0)       0
ACTION Idling(D: 0.1, M:0 )                      0          0.2(1) 0.09999979(-0.9)           0(1)            0(0)            0(0)            0(0)        0
ACTION Wandering(D: -0.1, M:0 )                  0         -0.2(1) 0.09999979(0.9)           0(1)            0(0)            0(0)            0(0)       0
ACTION Woodcutting(D: -0.4, M:0 )                0          0.2(1)  -0.3000001(1)         -0.1(1)            0(0)            0(0)            0(0)       0
ACTION Mining(D: -0.4, M:0 )                     0          0.2(1)  -0.3000001(1)         -0.1(1)            0(0)            0(0)            0(0)       0
ACTION Fishing(D: -0.4, M:0 )                    0          0.2(1)  -0.3000001(1)         -0.1(1)            0(0)            0(0)            0(0)       0
ACTION ShopKeeper(D: -0.4, M:0 )                 0          0.2(1)  -0.3000001(1)         -0.1(1)            0(0)            0(0)            0(0)       0
ACTION PlayingMusic(D: -0.4, M:0 )               0          0.2(1)  -0.3000001(1)         -0.1(1)            0(0)            0(0)            0(0)       0
ACTION Hunting(D: -1.1, M:0 )                    0       -0.2(0.8)  -0.3000001(1)         -0.1(1)            0(0)            0(0)            0(0)       0
ACTION Farming(D: -1.1, M:0 )                    0       -0.2(0.8)  -0.3000001(1)         -0.1(1)            0(0)            0(0)            0(0)       0
ACTION Blacksmithing(D: -1.1, M:0 )              0       -0.2(0.8)  -0.3000001(1)         -0.1(1)            0(0)            0(0)            0(0)       0
ACTION Drinking(D: -1.3, M:0 )                   0         0.2(-2)   0.09999979(1)            0(1)           -1(0)            0(0)           -1(0)      -1
ACTION Fighting(D: -1.8, M:0 )                  -1            0(1)   -0.6999999(1)         -0.1(1)           0(-2)            0(0)            0(0)       0
```

Figure 4.4: Detailed information about the choice of actions the NPC takes.

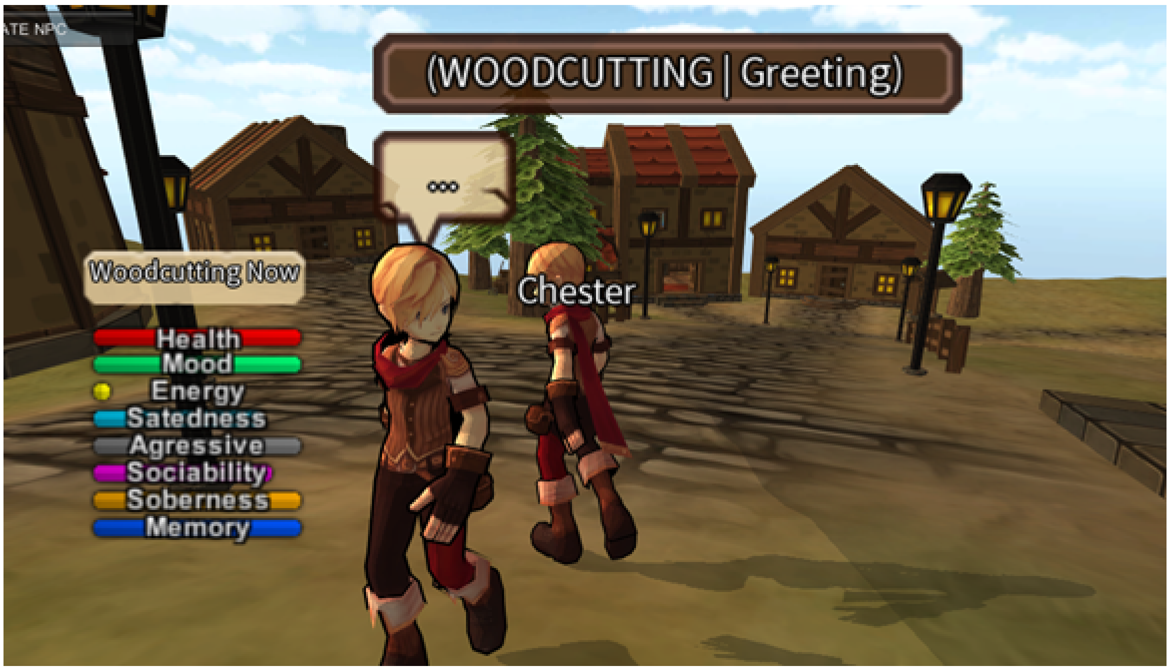
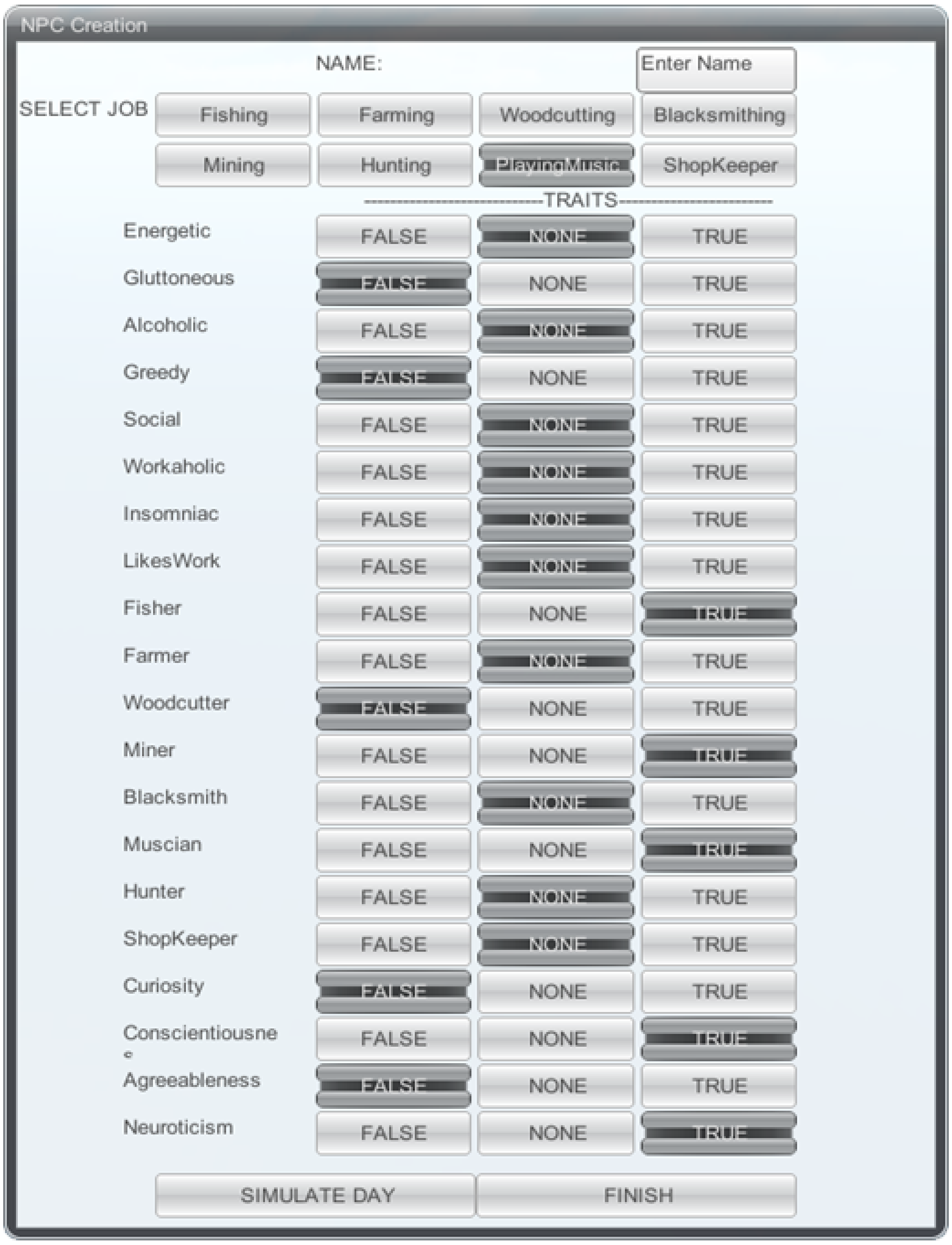


Figure 4.5: Figure showing a NPC with the emotion model debug information displayed.


NPC Creation

NAME: Enter Name

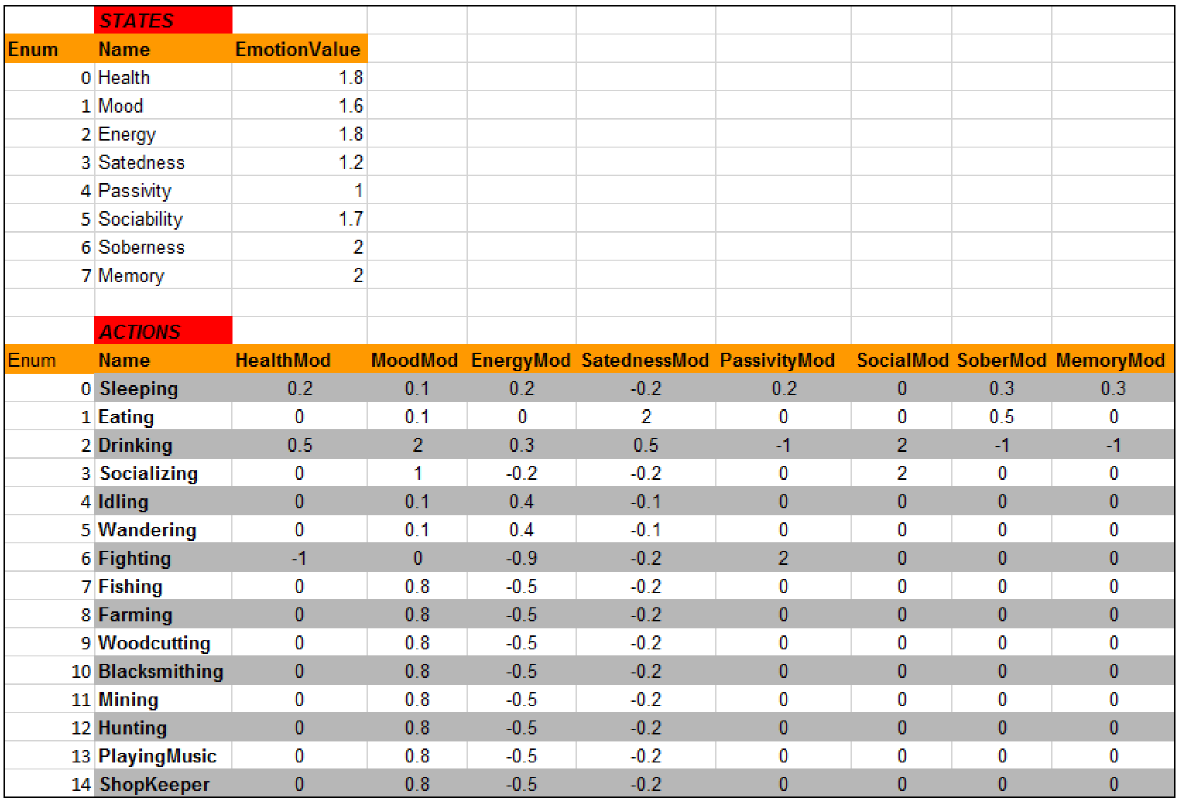
SELECT JOB: Fishing Farming Woodcutting Blacksmithing Mining Hunting [✓ PlayingMusic] ShopKeeper

-----TRAITS-----

Energetic	FALSE	[✓ NONE]	TRUE
Gluttoneous	[✓ FALSE]	NONE	TRUE
Alcoholic	FALSE	[✓ NONE]	TRUE
Greedy	[✓ FALSE]	NONE	TRUE
Social	FALSE	[✓ NONE]	TRUE
Workaholic	FALSE	[✓ NONE]	TRUE
Insomniac	FALSE	[✓ NONE]	TRUE
LikesWork	FALSE	[✓ NONE]	TRUE
Fisher	FALSE	NONE	[✓ TRUE]
Farmer	FALSE	[✓ NONE]	TRUE
Woodcutter	[✓ FALSE]	NONE	TRUE
Miner	FALSE	NONE	[✓ TRUE]
Blacksmith	FALSE	[✓ NONE]	TRUE
Muscian	FALSE	NONE	[✓ TRUE]
Hunter	FALSE	[✓ NONE]	TRUE
ShopKeeper	FALSE	[✓ NONE]	TRUE
Curiosity	[✓ FALSE]	NONE	TRUE
Conscientiousne e	FALSE	NONE	[✓ TRUE]
Agreeableness	[✓ FALSE]	NONE	TRUE
Neuroticism	FALSE	NONE	[✓ TRUE]

SIMULATE DAY FINISH

Figure 4.6: Image showing the dialogue box used when creating a NPC during run time.



	STATES	
Enum	Name	EmotionValue
0	Health	1.8
1	Mood	1.6
2	Energy	1.8
3	Satedness	1.2
4	Passivity	1
5	Sociability	1.7
6	Soberness	2
7	Memory	2

	ACTIONS								
Enum	Name	HealthMod	MoodMod	EnergyMod	SatednessMod	PassivityMod	SocialMod	SoberMod	MemoryMod
0	Sleeping	0.2	0.1	0.2	-0.2	0.2	0	0.3	0.3
1	Eating	0	0.1	0	2	0	0	0.5	0
2	Drinking	0.5	2	0.3	0.5	-1	2	-1	-1
3	Socializing	0	1	-0.2	-0.2	0	2	0	0
4	Idling	0	0.1	0.4	-0.1	0	0	0	0
5	Wandering	0	0.1	0.4	-0.1	0	0	0	0
6	Fighting	-1	0	-0.9	-0.2	2	0	0	0
7	Fishing	0	0.8	-0.5	-0.2	0	0	0	0
8	Farming	0	0.8	-0.5	-0.2	0	0	0	0
9	Woodcutting	0	0.8	-0.5	-0.2	0	0	0	0
10	Blacksmithing	0	0.8	-0.5	-0.2	0	0	0	0
11	Mining	0	0.8	-0.5	-0.2	0	0	0	0
12	Hunting	0	0.8	-0.5	-0.2	0	0	0	0
13	PlayingMusic	0	0.8	-0.5	-0.2	0	0	0	0
14	ShopKeeper	0	0.8	-0.5	-0.2	0	0	0	0

Figure 4.7: CSV File used for emotional states and action information.

Chapter 5

Evaluation

This chapter is concerned with evaluating the Emotional Model created. The question of whether the NPCs created are realistic or not is answered, following this a complexity analysis of the action selection manager is performed. Following this using Unity3D's profiler some values are mentioned about the efficiency of the code.

Background AI characters must operate with little use of memory and CPU requirements so that these resources can be used for the more essential features in a game environment. There are three main areas we can evaluate for an AI model, *CPU cost*, *Memory cost* and *Character Fidelity*. CPU cost and memory cost can be directly measured as but there is no metric for character fidelity, there is no clear way to judge whether an AI agent is realistic other than through testing and user studies.

5.1 Believable NPCs

The main objective of this dissertation is to increase the believability of NPC's (1.3.1). The question of "Is this NPC believable?" can be difficult to answer. Due to the fact that the action selection manager has a slight bit of randomness involved, it can be difficult to discover if the NPCs action choices are believable, or a little lucky. It is also down to personal judgement whether the NPC's schedule is correct or not. Below the schedules of some example NPCs are shown which are plausible human behaviour.

They sleep during the night, they work while taking the occasional breaks to eat or relax if they get tired, and during the evening they relax or preform a pastime or drink. It is easy to decide whether the NPC's choices are correct with knowledge of the NPC's traits, but in a game it might be difficult for the player to understand unless there is a dialogue system where the player can talk to the NPC to learn about his traits. This could be an interesting mechanic to add to the game. Games such as Skyrim already have a dialogue system in place where you can ask the NPCs for more information about themselves. This would provide the player with a little more information and would add the the immersion of the game that these NPCs have unique personalities the player can learn about.

Below is an example of an NPC's simulated day. The tables 5.1 and 5.2 show the contents of the text file that is exported when the model is run. For easy reading it is exported to a table format and the main points are outlined. In table 5.1 the name and job of the NPC are shown, as well as the random traits selected. *Neuroticism* makes the NPC angry whenever he works and since he is not *Gluttonous* he will be less likely to eat as much. Likewise because he is not an *Alcoholic* he will not drink as much. Table 5.2 shows the two simulated days for this NPC. The key emotions to look out for is the passivity, energy and satedness of the NPC. For readability only values that have changed are shown, if the value is the same as the previous hour a "-" is used. Since this NPC likes his job, there is very little that upsets his mood so it rarely changes, however, seeing as he works as a blacksmith it makes him a little angry. This causes him to eventually fight to relieve his stress. This in turn injures the NPC's health, which is regenerated through sleeping or drinking.

As mentioned in the implementation, the ability to print all the information from the NPCs is helpful for debugging them. This data can be exported into Excel, and a graph of the NPC's life can be visualized. With the use of this it can be easier to try judge the NPC's life and determine if it is plausible in accordance to the NPC's traits. It also helps visualize the emotional state values as they are adjusted after each action. Figures 5.1 [p.60] and 5.2 [p.61] are graphs showing a simulated day for two NPCs created by the Emotional Model.

Figure 5.1 on page 60 presents an NPC who is a blacksmith. Some traits of his are that he likes his job but it makes him angry, he is not an alcoholic, he likes to be

Name	**Job**
Yuri	Blacksmith
Traits	**T or F**
Gluttonous	F
Alcoholic	F
Social	T
Likes Work	T
Woodcutter	F
Miner	F
Hunter	T
Neuroticism	T

Table 5.1: Yuri the Blacksmith, an NPC created by the Model.

social and likes to hunt. The graph plots the emotional states against the actions the NPC chooses. This is simulated for 3 days. The important parts of the graph are the energy, satedness and passivity states of the NPC. A clear pattern is observed such as the NPC getting tired during the day, however, his actions for all three days are slightly different. While there is a predictable element as to what actions the NPC will choose, it is a little less predictable when he will choose them.

Another NPC is presented in Figure 5.2 on page 61. This NPC who is a woodcutter really dislikes his job, he is an alcoholic, he likes to play music and fish. The graph presented is a little more cluttered and can be a little difficult to understand. The main states to look out for is the NPC's mood which changes whenever he works because of his dislike for it. This causes him to seek out something more pleasant which on day two ends up with him drinking, a lot.

Looking at these two NPCs, who were created randomly from the model, they appear to have believably lives.

Hour	**Action Selected**	**Health**	**Mood**	**Energy**	**Satedness**	**Passivity**	**Sociability**	**Soberness**	**Memory**
0	Sleeping	2.0	1.7	2.0	1.1	1.2	1.7	2.0	2.0
1	Eating	-	1.8	-	2.0	-	-	-	-
2	Wandering	-	1.9	-	-	-	-	-	-
3	Wandering	-	2.0	-	-	-	-	-	-
4	Sleeping	-	-	-	1.9	1.4	-	-	-
5	Sleeping	-	-	-	1.8	1.6	-	-	-
6	Sleeping	-	-	-	1.7	1.8	-	-	-
7	Sleeping	-	-	-	1.6	2.0	-	-	-
8	Sleeping	-	-	-	1.5	-	-	-	-
9	Blacksmithing	-	-	1.5	1.4	1.8	-	-	-
10	Blacksmithing	-	-	1	1.3	1.6	-	-	-
11	Blacksmithing	-	-	0.5	1.2	1.4	-	-	-
12	Socializing	-	-	0.3	1.1	-	2.0	-	-
13	Idling	-	-	0.7	-	-	-	-	-
14	Idling	-	-	1.1	-	-	-	-	-
15	Blacksmithing	-	-	0.6	1.0	1.2	-	-	-
16	Eating	-	-	-	2.0	-	-	-	-
17	Idling	-	-	1.0	-	-	-	-	-
18	Idling	-	-	1.4	-	-	-	-	-
19	Idling	-	-	1.8	-	-	-	-	-
20	Fighting	1.0	-	0.9	1.9	2.0	-	-	-
21	Wandering	-	-	1.3	-	-	-	-	-
22	Idling	-	-	1.7	-	-	-	-	-
23	Sleeping	1.2	-	1.9	1.8	-	-	-	-
0	Wandering	-	-	2.0	-	-	-	-	-
1	Sleeping	1.4	-	-	1.7	-	-	-	-
2	Sleeping	1.6	-	-	1.6	-	-	-	-
3	Sleeping	1.8	-	-	1.5	-	-	-	-
4	Sleeping	2.0	-	-	1.4	-	-	-	-
5	Sleeping	-	-	-	1.3	-	-	-	-
6	Sleeping	-	-	-	1.2	-	-	-	-
7	Sleeping	-	-	-	1.0	-	-	-	-
8	Sleeping	-	-	-	-	-	-	-	-
9	Blacksmithing	-	-	1.5	0.9	1.8	-	-	-
10	Blacksmithing	-	-	1.0	0.8	1.6	-	-	-
11	Blacksmithing	-	-	0.5	0.7	1.4	-	-	-
12	Blacksmithing	-	-	0	0.6	1.2	-	-	-
13	Fighting	1.0	-	-	0.5	2.0	-	-	-
14	Drinking	1.5	-	0.3	1.1	1.0	-	1.0	1.0
15	Wandering	-	-	0.7	-	-	-	-	-
16	Blacksmithing	-	-	0.2	1.0	0.8	-	-	-
17	Wandering	-	-	0.6	-	-	-	-	-
18	Idling	-	-	1.0	-	-	-	-	-
19	Wandering	-	-	1.4	-	-	-	-	-
20	Idling	-	-	1.8	-	-	-	-	-
21	Drinking	2.0	-	2.0	1.6	0.0	0.0	0.0	0.0
22	Fighting	1.0	-	1.1	1.5	2.0	2.0	-	-
23	Sleeping	1.2	-	1.3	1.4	-	-	0.3	0.3
0	Sleeping	1.4	-	1.7	1.3	-	-	0.6	0.6

Table 5.2: 48 hr. simulation of NPC with traits from 5.1.

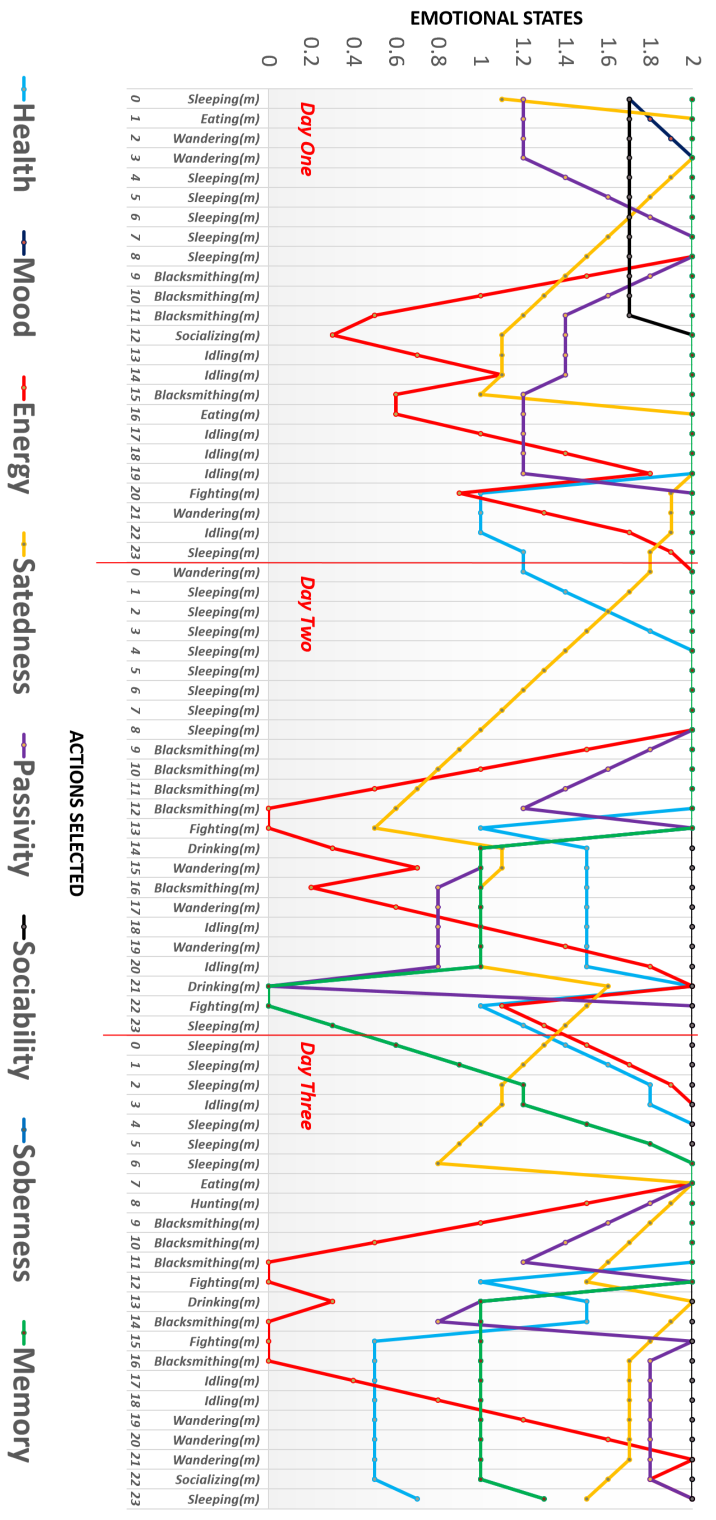


Figure 5.1: NPC Example1. This blacksmith likes his job, is not an alcoholic, likes to be social, likes to hunt , gets angry when he works.

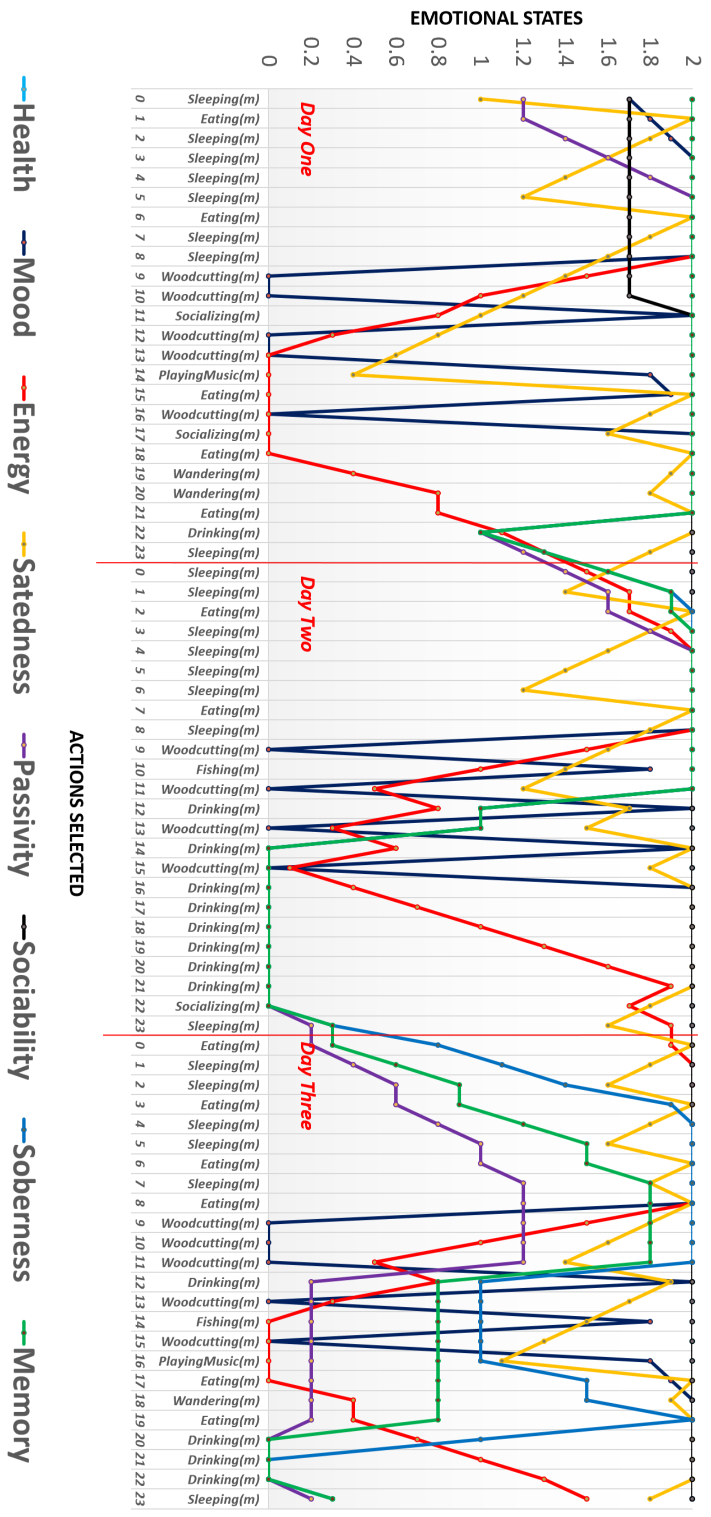


Figure 5.2: NPC Example2. This woodcutter really dislikes his job, is an Alcoholic, he likes to fish and play music and dislikes every other profession.

Function	**Size**	**Complexity Avg.**	**Complexity Worst case**
For-each Loop	# of Actions	A * S	A * S
Addition to list	# of Actions	D	A
List sorting	# of Deltas	$O(D^2)$	$O(A^2)$
For loop	4 elements	4	D
For loop	4 elements	4	D

Table 5.3: Complexity Analysis of Action Selection Manager. A is the number of Actions, S is the number of emotional states and D is the number of deltas.

5.2 Complexity Analysis

The ability of the model to scale can be observed in table 5.4, which shows a time analysis of the model. Figure 5.3 shows a breakdown of the main functions inside the Action Selection Manager. For this table, N is the number of NPCs, A is the number of actions, S is the number of states and D is the number of delta values. During the update the number of delta values, D, will be the only non constant value that will change during each update. Table 5.3 is for each NPC, this is the action selection manager which is run each update.

Scaling for each NPC is $O(A^2)$ where A is the number actions. This is due to a single insertion sort in the middle of the Action Selection Manager. The delta list contains a maximum of the number of actions, but is generally less than this as the delta is omitted if it is 0. This is the case for sleeping and working when the activity times are 0.

5.3 Runtime Performance

The model was implemented and tested in Unity3D as previously mentioned, using Unity3Ds profiler, the model was run in several different scenarios. Firstly the model was run in an isolated environment with only 2 NPCs and with debug information on, meaning that it printed to a text file about the NPCs live. The second run was the same except with all debug information off. The third run was 60+ NPCs updating at the same time with no debug information. In table 5.4 it can be seen that the

Test	**# of NPCs**	**CPU Time (ms)**
With debug	2	30.32
Without debug	2	0.66
Without debug	60+	2.84
Without debug	120+	5.76
Without debug	240+	11.87

Table 5.4: CPU Usage Time

printing of debug information has a huge effect on the model, however, without any debug information, the model runs and preforms efficiently. It has very little impact with 60+ NPCs only taking 2.84 ms to update all at once, and this was done without staggering the updates of the NPCs. Other tests doubling the size of NPCs were also conducted and shown below. While these numbers can be a little extreme they can be used as an indicator to see the scaling of the model.

For more efficiency since the NPCs actions are picked every hour, it would be possible to stagger the updates to try minimize the performance required. This is preformed using the NPC manager (see 4.3.3).

Chapter 6

Conclusion

The purpose of this chapter is to summarise of the contributions made by this dissertation and make a final comment on the potential future for a model such as this. A brief overview is discussed as well as some potential future work and ending with final thoughts. The original goals are also presented and their realisation evaluated.

6.1 Contributions

This section mentions the contributions that this dissertation has had.

While this dissertation was built on top of a model designed by John Fallon last year, some improvements have been made that overall increases the fidelity of these NPCs. This model was fully implemented in Unity3D with little impediments.

It was demonstrated that adding this model to existing NPCs in a game environment was done with relative ease shown in section 4.3.1. The potential of this model has been demonstrated with numerous example NPCs created, tending to have believable lives (5.1). The way the model is designed allows other developers to easily extend the model to their liking, whether they require the NPCs to be procedurally generated (3.9) or customized (4.3.2). Both of these methods for creating the NPCs were objectives stated towards the start of this dissertation 1.3.2 and 1.3.3.

The model was hooked up with the NPC's model to visually represent how the NPC is currently feeling 4.3.1. This was a requirement stated by Loyall[20] that the "Consistency of Expression" of the character's emotions and animation.

6.2 Future Work

While the Emotional Model was fully implemented and demonstrated, there were a few areas where it lacked proper attention. The temporary removal of the external factors 3.7, which were not reintroduced by the end, could be reimplemented. Much would be required attempting to balance these values so they do not effect the NPC too much.

Currently the initial excel file 4.3.4, used to read in the emotional states and actions, could be further improved. One improvement would be the addition of the traits into this file so that, in order for a developer to add in custom traits it would only require changing this excel sheet rather than having to code.

General improvements in efficiency for the model could be preformed. The complexity of the model is $O(A^2)$ (see 5.2) due to an insertion sort. This could be improved to make the model more efficient at scaling.

The multipliers (3.4.3) could be further improved. As these were added quiet late in the development, they were not fully balanced and it is believed that they could be a huge factor in improving the believability of the NPCs.

More work could be done on the action selection manager, since it is the main component of the model. A more sophisticated probability model could be implemented for deciding which action to choose, perhaps adding weights to certain actions if they require priority.

The game environment used to test the model only had 13 NPCs and, while the model was run with 60+ NPCs concurrently (5.3), further testing for use in a open world game would be required.

Since traits are a big part of the model for giving the NPCs personalities, the addition of more traits could give NPCs even more characteristics and variety.

More dynamic events could be added to alter the NPC's states such as a day night cycle, or the addition of sickness which would alter the NPC's multipliers 3.7.

A slightly altered version of this emotion model could be used for animal behaviour: with a few changes in actions and a limitation of traits, the model could be used effectively for animals in a open world game.

6.3 Final Thoughts

The model proposed has the potential for use in games. Further testing would be required before use in a open world game like Skyrim. The model was shown to be efficient and effective at creating NPCs which are believable (see 5.1), which is the main goal of this dissertation (1.3.1). With the relative ease of implementing the model to a game environment, it could be attractive to game developers. Since the model is coded in C# it could be easily exported for use in other game engines and easily expanded with the ideas mentioned in future work.

Appendix

Appendix A: Source code on CD/DVD included.

Bibliography

[1] *Grand Theft Auto*. Rockstar Games, 1998.

[2] *Grand Theft Auto V*. Rockstar Games, 2013.

[3] *The Elder Scrolls V: Skyrim*. Bethesda Game Studios, 2011.

[4] *S.T.A.L.K.E.R.: Shadow of Chernobyl*. GSC Game World, 2007.

[5] *World Of Warcraft*. Blizzard Entertainment, 2004.

[6] *Guild Wars 2*. ArenaNet, 2012.

[7] *Second Life*. Linden Research, Inc, 2003.

[8] *Aion: The Tower of Eternity*. NA NCsoft, 2008.

[9] *RuneScape*. Jagex Games Studio, 2001.

[10] *Lineage II*. NCSOFT, 2003.

[11] *Pac-Man*. Namco, 1980.

[12] C. Birch, "Understanding pac-man ghost behavior," December 2010. $http : //gameinternals.com/post/2072558330/understanding - pac - man - ghost - behavior$.

[13] D. L. Roberts, M. O. Riedl, and C. L. Isbell, "Beyond adversarial: The case for game ai as storytelling."

[14] *Half-Life 2*. Valve, 2004.

[15] *Bioshock Infinite*. Irrational Games, 2013.

[16] *The Walking Dead.* Telltale Games, 2012.

[17] *The Wolf Among Us.* Telltale Games, 2013.

[18] W. Ali and B. Moulin, "How artificial intelligent agents do shopping in a virtual mall: A believable and usable multiagent-based simulation of customers shopping behavior in a mall," in *Advances in Artificial Intelligence* (L. Lamontagne and M. Marchand, eds.), vol. 4013 of *Lecture Notes in Computer Science*, pp. 73–85, Springer Berlin Heidelberg, 2006.

[19] C. Pedica and H. Vilhjlmsson, "Spontaneous avatar behavior for human territoriality," in *Intelligent Virtual Agents* (Z. Ruttkay, M. Kipp, A. Nijholt, and H. Vilhjlmsson, eds.), vol. 5773 of *Lecture Notes in Computer Science*, pp. 344–357, Springer Berlin Heidelberg, 2009.

[20] A. B. Loyall, *Believable Agents: Building Interactive Personalities.* Ph.d. thesis, Carnegie Mellon University, Pittsburgh, PA, USA, May 1997.

[21] R. W. Picard, *Affective Computing.* Cambridge, MA, USA: MIT Press, 1997.

[22] M. P. Eladhari, H. Verhagen, J. McCoy, and M. Johansson, "Social believability in games." *https : //sites.google.com/site/socialbelievabilityingames/.*

[23] J. Dimas and R. Prada, "Social behaviour in games: Dynamic identity in npcs," in *Workshop on Social Believability in Games at The 10th International Conference on Advances in Computer Entertainment Technology (ACE)*, ACM, November 2013.

[24] M. Bruijnes, J. Kolkmeier, R. op den Akker, J. Linssen, M. Theune, and D. Heylen, "Keeping up stories: design considerations for a police interview training game," in *Social Believability in Games Workshop, SBG 2013*, (Enschede, The Netherlands), University of Twente, Centre for Telematics and Information Technology, November 2013.

[25] H. Ahn and R. W. Picard, "Affective cognitive learning and decision making: The role of emotions," in *The 18th European Meeting on Cybernetics and Systems Research (EMCSR 2006)*, pp. 1–6, 2006.

[26] C. Gershenson, "Modelling emotions with multidimensional logic," in *Fuzzy Infor-*

mation Processing Society, 1999. NAFIPS. 18th International Conference of the North American, pp. 42–46, Jul 1999.

[27] R. Prada, J. a. Camilo, and M. A. Nunes, “Introducing personality into team dynamics,” in *Proceedings of the 2010 Conference on ECAI 2010: 19th European Conference on Artificial Intelligence*, (Amsterdam, The Netherlands, The Netherlands), pp. 667–672, IOS Press, 2010.

[28] R. R. McCrae and O. P. John, “An introduction to the five-factor model and its applications,” *Journal of Personality*, vol. 60, no. 2, pp. 175–215, 1992.

[29] G. Acampora, F. Ferraguto, and V. Loia, “Synthesizing bots emotional behaviors through fuzzy cognitive processes,” in *Computational Intelligence and Games (CIG), 2010 IEEE Symposium on*, pp. 329–336, Aug 2010.

[30] J. E. Laird and M. van Lent, “Human-level ai’s killer application: Interactive computer games,” 2000.

[31] R. Alur and D. L. Dill, “A theory of timed automata,” *Theoretical Computer Science*, vol. 126, pp. 183–235, 1994.

[32] E. Games, *Unreal Tournament 2004*. Atari, Inc., 2004.

[33] A. Ortony, G. Clore, and A. Collins, *Cognitive Structure of Emotions*. Cambridge University Press, 1988.

[34] W. Blewitt, A. Ayesh, C. Bertelle, and K. Mahboub, “Psychologically grounded emotion model for a npc fuzzy controller,” ., 2012.

[35] J. Millenson, “The psychology of emotion: Theories of emotion perspective,” ., pp. 35–36, 1967.

[36] MathWorks, “Fuzzy.” $http://www.mathworks.co.uk/help/fuzzy/fuzzy.html$.

[37] CADIA, “Cadia populus.” $http://populus.cs.ru.is/$.

[38] C. CADIA, “Humanoid agents in social game environment.” $http://cadia.ru.is/wiki/public:socialgame:main$.

[39] CADIA, “Cadia.” $http://cadia.ru.is/$.

[40] *Zork*. Infocom, 1977.

[41] *Thief: The Dark Project*. Looking Glass Studios, 1998.

[42] *Black & White*. Lionhead Studios, 1998.

[43] *FableII*. Lionhead Studios, 2008.

[44] I. Umarov and M. Mozgovoy, "Believable and effective ai agents in virtual worlds: Current state and future perspectives.," *IJGCMS*, vol. 4, no. 2, pp. 37–59, 2012.

[45] I. Algorithm, "Lydia vs. the gate : Will she ever learn?," November 2011.

[46] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *Systems Science and Cybernetics, IEEE Transactions on*, vol. 4, pp. 100–107, July 1968.

[47] W. G. van Toll, A. F. Cook, and R. Geraerts, "A navigation mesh for dynamic environments," *Comput. Animat. Virtual Worlds*, vol. 23, pp. 535–546, Nov. 2012.

[48] A. Blog, "Fixing pathfinding once and for all," July 2008. $http://www.ai-blog.net/archives/000152.html$.

[49] A. Johansson, "Affective decision making in articial intelligence," Master's thesis, Linkping University, 2012.

[50] *Neverwinter Nights*. BioWare, 2002.

[51] R. E. Fikes and N. J. Nilsson, "Strips: A new approach to the application of theorem proving to problem solving," in *Proceedings of the 2Nd International Joint Conference on Artificial Intelligence*, IJCAI'71, (San Francisco, CA, USA), pp. 608–620, Morgan Kaufmann Publishers Inc., 1971.

[52] J. Orkin, "Three states and a plan: The ai of f.e.a.r." $http://web.media.mit.edu/\ jorkin/gdc2006_orkin_jeff_fear.pdf$.

[53] *F.E.A.R.* Monolith Productions, 2007.

[54] E. Long, "Enhanced npc behaviour using goal oriented action planning," Master's thesis, University of Abertay Dundee, 2007. $http://www.edmundlong.com/Projects/Masters_EnhancedBehaviourGOAP_EddieLong.pdf$.

[55] *Halo.* Bungie, 2001.

[56] *Saints Row The Third.* THQ, 2011.

[57] J. O. J, "Ai planning." $http://jiyambi.blogspot.ie/2012/11/ai-planning.html$.

[58] A. Johansson and P. Dell'Acqua, "Emotional behavior trees," in *Computational Intelligence and Games (CIG), 2012 IEEE Conference on*, pp. 355–362, Sept 2012.

[59] *The Last Of Us.* Naughty Dog, 2014.

[60] A. Champandard, "Game ai innovation in fable 2," 2007. $http://aigamedev.com/open/article/innovation-fable-2/$.

[61] *Faade.* Michael Mateas and Andrew Stern, 2005.

[62] P. Gorman, "Dramatic structure." $http://paulgorman.org/writing/dramatic_structure.php$.

[63] *Just Cause 2.* Avalanche Studios, 2010.

[64] A. Champandard, "Planning in games: An overview and lessons learned." $http://aigamedev.com/open/review/planning-in-games/$.

[65] A. Champandard, "A-life, emergent ai and s.t.a.l.k.e.r.: An interview with dmitriy iassenev." $http://aigamedev.com/open/interviews/stalker-alife/$.

[66] "Radiant a.i." $http://elderscrolls.wikia.com/wiki/Radiant_A.I.$.

[67] *The Elder Scrolls IV: Oblivion.* Bethesda Game Studios, 2006.

[68] *The Elder Scrolls III: Morrowind.* Bethesda Game Studios, 2002.

[69] J. Fallon, "Believable behaviour of background characters in open world games," Master's thesis, Trinity College Dublin, 2013.

[70] *Half-Life.* Valve, 1998.

[71] S. A. Ian Patterson and P. Connelly, "Skyrim script extender." $http://skse.silverlock.org/$.

[72] T. McNulty, "Residual memory for background characters in complex environments," Master's thesis, Trinity College Dublin, 2014.

[73] U. Technologies, “Unity3d game engine.” $http : //unity3d.com/$.