PERSONALITY AND MOOD FOR NON-PLAYER CHARACTERS: A METHOD FOR BEHAVIOR SIMULATION IN A MAZE ENVIRONMENT

A Thesis

presented to

the Faculty of California Polytechnic State University,

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Computer Science

by

Noah Paige

COMMITTEE MEMBERSHIP

TITLE: Personality and Mood for Non-Player Characters: A Method for Behavior Simulation in a Maze Environment

AUTHOR: Noah Paige

DATE SUBMITTED: December 2020

COMMITTEE CHAIR: Christian Eckhardt, Ph.D.
Professor of Computer Science

COMMITTEE MEMBER: Foaad Khosmood, Ph.D.
Professor of Computer Science

COMMITTEE MEMBER: Franz Kurfess, Ph.D.
Professor of Computer Science

ABSTRACT

Personality and Mood for Non-Player Characters: A Method for Behavior Simulation in a Maze Environment

Noah Paige

When it comes to video games, immersion is key. All types of games aim to keep the player immersed in some form or another. A common aspect of the immersive world in most role-playing games – but not exclusive to the genre – is the non-playable character (NPC). At their best, NPCs play an integral role to the sense of immersion the player feels by behaving in a way that feels believable and fits within the world of the game. However, due to lack of innovation in this area of video games, at their worst NPCs can jar the player out of the immersive state of flow with unnatural behavior.

In an effort towards making non-playable characters (NPCs) in games smarter, more believable, and more immersive, a method based in psychological theory for controlling the behavior of NPCs was developed. Based on a behavior model similar to most modern games, our behavior model for NPCs traverses a behavior tree. A novel method was introduced using the five-factor model of personality (also known as the big-five personality traits) and the circumplex model of affect (a model of emotion) to inform the traversal of the behavior tree of NPCs. This behavior model has two main beneficial outcomes. The first is emergent gameplay, resulting in unplanned, unpredictable experiences in games which feel closer to natural behavior, leading to an increase in immersion. This can be used for complex storytelling as well by offering information about an NPC's personality to be used in the narrative of games. Secondly, the model is able to provide the emotional status of an NPC in real time. This capability allows developers to programmatically display facial and body expression,

eschewing the current time-consuming approach of artist-choreographed animation. Finally, a maze simulation environment was constructed to test the results of our behavior model and procedural animation.

The data collected from 100 iterations in our maze simulation environment about our behavior model found that a correlation can be observed between traits and actions, showing that emergent gameplay can be achieved by varying personality traits. Additionally, by incorporating a novel method for procedural animation based on real-time emotion data, a more realistic representation of human behavior is achieved.

ACKNOWLEDGMENTS

Thanks to:

- My family, for supporting me
- Sydney, for your encouragement and being the best rubber ducky
- Professor Eckhardt, for pushing me to pursue my dreams

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

"If you can't tell the difference, does it matter if I'm real or not?"[32], asks the artificially intelligent 'host' in *Westworld*. In a show about a theme park filled with AI human facsimiles where there are no rules or consequences – and, more broadly, about the artificial intelligence singularity – the line between what is real and artificial is blurred. While current technology hasn't yet reached the problematically lofty heights of *Westworld*, today's video games share the common goal of blurring the line between real and artificial.

When game designers talk about blurring the line, they typically use the word immersion. An immersive game is one that seizes the player's attention and makes them feel like the game they are playing is real. Game developers strive to achieve immersion in different ways, which vary based on the type of game that they are working on. For instance, a racing game would likely focus on car physics, a fighting game would work on pixel-perfect animations, or a role playing game on a rich, believable world to explore. One part of creating a rich, believable world in role playing games – but not exclusive to the role playing genre – is the people that inhabit it.

For games, the allegorical counterpart to Westworld's 'hosts' are non-player characters (NPCs). They are an important, yet oft-neglected component of realism and immersion in games. Many games have made advances in depicting their NPCs in regards to graphics or animation in popular games, but the underlying artificial intelligence has been mostly unchanged for decades.

Johansson and Warpefelt (2013)[43] explain that most problems with NPCs in games are due to their unbelievability, particularly in social interactions. While other aspects of a game may contribute to blurring the line of reality, today's NPCs usually have the opposite effect. They often exhibit unexpected and unbelievable behavior within the context of any given game, resulting in a loss of immersion.

In this paper, we will discuss an NPC behavioral model, a method for procedural animation, and a maze simulation environment. Our behavioral model attempts to create more believable NPCs in games by incorporating the personality traits of the five-factor model [12] and the circumplex model of affect (a mood model). A byproduct of our behavior model is a dynamic mood for NPCs, and our method for procedural animation capitalizes on this dynamic mood by relating blendshapes to positions on our mood model. While the hosts of *Westworld* [32] attempt to solve a symbolic maze in their quest towards sentience, the behavior of our NPCs will be measured in a literal maze. A maze simulation environment was constructed in the Unity [4] game engine to test our behavioural model and our method for procedural animation. Through data collection in our maze simulation environment, we were able to measure a correlation between personality traits and changes in behavior.

Chapter 2

BACKGROUND

The following subsections will further explain the role of NPCs in video games, describe the AI behind NPCs, and introduce the Five-Factor model of personality, the circumplex model of affect, and procedural animation.

2.1 The Role of NPCs in Games

The term NPC first appeared not in video games, but in tabletop Role-Playing Games (RPGs), where they are controlled by a human Dungeon Master [27]. Within the scope of video games, non player characters (NPCs) are all the characters in a game that are not controlled by a human, and instead are controlled by the computer. In video games, the role of Dungeon Master is played by the artificial intelligence that controls NPC behavior. While video games and tabletop RPGs differ in the medium used, NPCs between the two still share quite a bit of similarities.

In video games, just like in tabletop RPGs, NPCs can take on many different appearances. Depending on the setting of the game, an NPC could be humanoid or non humanoid in shape. Despite the drastic variance in appearance of NPCs, their roles are quite consistent across games.

Warpefelt [42] provides a comprehensive typology of the roles that NPCs play in games. Warpefelt lists the main four classifications for an NPC to be: functions, adversaries, friends, and providers. Function NPCs do things like give quests, provide services, or act as vendors. Adversary NPCs take the form of enemies and bosses.

Friend NPCs take the form of sidekicks, companions, pets, and minions. Provider NPCs are storytellers and loot providers.

2.2 How do NPCs Work?

The NPCs of most popular AAA games are controlled with state machines, which are a very common method of controlling various software and hardware. In video games, state machines can be used to determine the behavior of AI agents. A state machine, at its core, is a directed graph. In an NPC state machine graph, nodes denote behavior states and edges denote events. A simple state machine example in a game is shown in Figure 2.1, which shows the state of an NPC guard in a fictional game setting. This state machine consists of two nodes, 'Standing Guard', and 'Attacking', and one edge, 'Player Seen'. In this example, the NPC will start in the 'Standing Guard' node and stay in that behavior state indefinitely unless they see a player. When a player is seen, the NPC will then transition to the 'Attacking' state. While it does well at illustrating how state machines work, his simple state machine is not typical for modern games in terms of complexity. A typical NPC state machine in a real game would likely have tens, maybe hundreds of states.

Figure 2.1: A simple state machine consisting of two states and a transition [16].

All of the many states and transitions within a state machine must be created by developers. Because of this predetermined design, NPCs are not very flexible in adapting to unexpected game scenarios, which are abundant in large scale games. When an NPC does not have a pre-designed transition/state pair for a given emergent gameplay scenario, the NPC must do its best and choose from the states it has, which often results in the player's sense of immersion being lost.

Trait	High Levels	Low Levels
Openness	Wide Interests Imaginitive Intelligent Original Insightful	Commonplace Narrow Interests Simple Shallow Unintelligent
Conscientiousness	Organized Thorough Planful Efficient Responsible	Careless Disorderly Frivolous Irresponsible Slipshot
Extraversion	Talkative Assertive Active Energetic Outgoing	Quiet Reserved Shy Silent Withdrawn
Agreeableness	Sympathetic Kind Appreciative Affectionate Soft-hearted	Fault-finding Cold Unfriendly Quarralsome Hard-headed
Neuroticism	Tense Anxious Nervous Moody Worrying	Stable Calm Contented Unemotional

Table 2.1: A list of commonly associated phrases for both high and low levels of each trait in the five-factor model. Phrases are listed in descending order of association to the given extreme for the trait. [20]

Trait	Facets
Openness	Ideas Fantasy Aesthetics Actions Feelings Values
Conscientiousness	Competence Order Dutifulness Achievement Striving Self-discipline Deliberation
Extraversion	Gregariousness Assertiveness Activity Excitement-seeking Positive Emotions
Agreeableness	Trust Straightforwardness Altruism Compliance Modesty Tender-mindedness
Neuroticism	Anxiety Angry Hostility Depression Self-consciousness Impulsiveness Vulnerability

Table 2.2: Costa & McCrae's NEO PI-R Facets. Six facets for each personality factor, where the facets for each trait provide a more detailed description for each trait [11].

2.3 The Five-Factor Model of Personality

A common representation of personality within the field of Psychology is the five-factor model [12]. To describe the personality of a particular person, each factor in the model is given a percentage value, where 0% indicates the person does not exhibit these characteristics at all, and 100% meaning they fully exhibit the characteristics.

The five factors are: openness, conscientiousness, extraversion, agreeableness, and neuroticism. For more explanation of each trait, refer to Table 2.1, which gives a good representation of each trait, listing commonly associated phrases for both high and low levels of each trait.

2.4 Circumplex Model of Affect

The circumplex model of affect is a model of emotional state that uses a two-dimensional graph to represent human emotion. This model proposes that all emotional states arise from cognitive interpretations of core neural sensations that are the product of two neurophysiological systems [34]. With this model, the two neurological systems, pleasantness and activation, are mapped to axes on the graph. Then, emotional states can be placed at specific points on the graph. See Figure 2.2 for an example of the graph, along with emotions labeled as points on the graph. For example, someone who exhibits high pleasantness and high arousal would put their position on the circumplex graph closest to the emotion of 'Delighted'.

2.5 Procedural Animation

In the games industry, character animation is typically achieved through motion capture or key framing that must be authored by artists [19]. Both methods require significant time and resources. Another little-used method for character animation, procedural animation, offers an algorithmic approach. Procedural animation is actually a quite broad term, but it can generally be defined as any animation that is achieved solely by an algorithm. The types of animation that fall into this broad category include things like water simulation (and other types of physics simulation) as well as simulation of life, like locomotion (walking, running, swimming, etc.), as

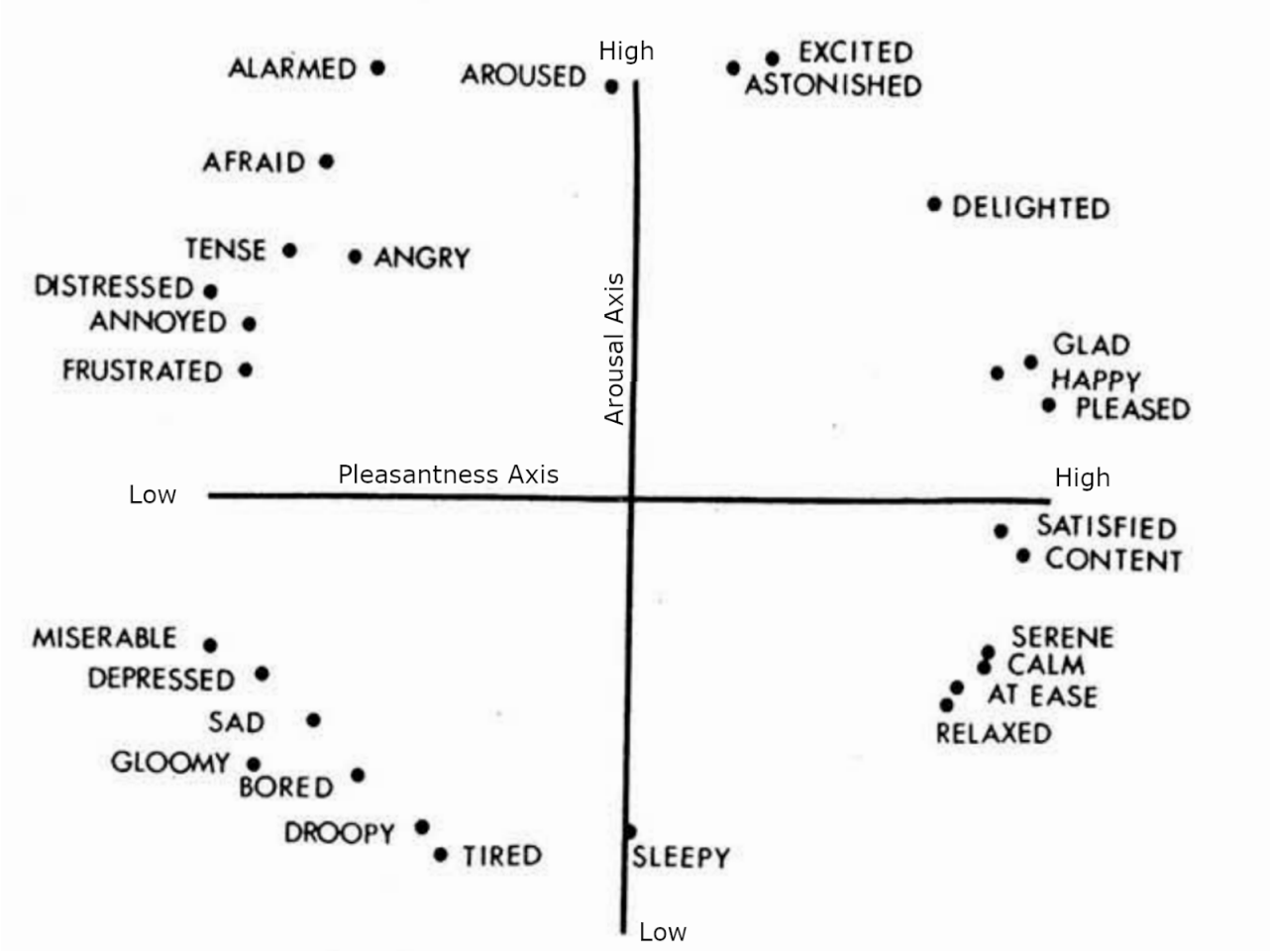

Figure 2.2: Russell's [37] circumplex model of affect showing 28 labeled positions of emotion on the graph, labeled by Russell. Edited to add axis and high/low labels.

well as emoting via facial expression. The focus of this work in regards to procedural animation is solely in relation to facial expression. However, chapter 8 expands on other types of procedural animation that could be used to enhance the emotive depth of NPCs.

The particular method for procedural facial animation that we use is known as blendshapes. First introduced in the computer graphics industry, blendshapes can be defined as linear facial models in which the individual basis vectors represent individual expressions [23]. More simply, a finite set of facial models are created for each char-

acter, with each model representing a specific facial expression. Then the artist or algorithm can blend between some combination of shapes in the set of facial models, effectively achieving an infinite set of expressions.

While blendshapes can be used for both procedural and hand-designed animation, this work will focus on using them in a procedural manner. Specifically, section 6.3 explains our use of blendshapes in conjunction with our mood model.

Chapter 3

RELATED WORK

In this chapter we will explore the various categories of work related to our topic and method of behavior simulation. We will cover research on personality and mood in NPCs, social believability, and drama management.

3.1 Personality And Mood in NPCs

Others have created similar approaches to improving NPCs through the use of Psychological principles and theory. Mac Namee [26] developed the first known approach, and drives NPC behavior through Eysenck's "two-dimensional classification" of personality [13] and Lang's "mood model" [22]. Eysenck's two-dimensional model of personality is limited compared to the five-factor model, with its two axes of extroversion/introversion and neuroticism/stability. Lang's mood model draws from Russel's [37] circumplex model of affect, and shares many similarities. Mac Namee's system has a static personality for each NPC, and a dynamic mood and perception of other characters. Mac Namee's system also uses trained artificial neural networks that are trained during development – before the players interact with the system.

Li and MacDonnell [24] developed an "emotion-base behavior model" that consists of several layers: the five-factor layer, the social layer, the emotion layer, and the interaction layer. Li and MacDonnell used the five-factor model of personality to represent an unchanging definition of personality for a character. Similar to Mac Namee's method, Li and MacDonnell's mood layer is the part of their system that changes over the course of gameplay. The emotion generator changes the emotional

state of the character through their interactions with a player, with three emotions being generated (shame, love, and shock) [15].

In their paper, NPCS AS PEOPLE, TOO: THE EXTREME AI PERSONALITY ENGINE [15], Georgeson credits their 2011 master's thesis, NPCs as People: Using Databases and Behaviour Trees to Give Non-Player Characters Personality [14] as the earliest work that, "utilized personality in a truly human way– that is, that included sophisticated base personalities that would develop and change over time based on the NPCs' lived experiences," [15]. While the original master's thesis is unpublished and inaccessible to the public, Georgeson's 2016 work NPCS AS PEOPLE, TOO: THE EXTREME AI PERSONALITY ENGINE provides a summary of the 2011 thesis.

In Georgeson's 2011 thesis, NPCs were given adaptive personalities that change over time based upon the input that represents interactions with the human-controlled player. Georgeson used the five-factor model to represent personality of NPCs, but in a different way than Li and MacDonnell. Georgeson's method uses all thirty facets of the five-factor model to describe personality (see Table 2.2). Additionally, Georgeson's method allows the personality of an NPC to change over time, which models psychological studies that show how a person's personality can change over the course of their life [33] [40] [38].

In their presentation at the 2012 Game Developer's AI Summit, Bura [9] describes a method for controlling NPC behavior that also uses the thirty facets of the five-factor model (see Table 2.2). In Bura's method, each facet is given a numerical score, and then many combinations of various facets are used to create NPC needs and behaviors.

3.2 Social Believability

In modern games, the believability of NPCs in general has been found to be lacking [17], but a great deal of research has been focused on fixing this issue with believability [41]. In the field of social believability, believability does not refer to truthfulness per se, but instead refers to whether a character seems lifelike, whether their actions make sense, and whether their behaviour suspends the player's sense of disbelief about their experience [41]. Bates provides another definition, saying that believable character is "one that provides the illusion of life, and thus permits the audience's suspension of disbelief" [8]. By this definition, a believable NPC is not necessarily one that perfectly resembles human behavior, but instead fits within their setting enough so that the player's immersion remains unbroken.

Chowanda et al. created study that explored the human player's experience in association with their ERiSA framework [10]. The ERiSA framework was used to imbue NPC companions with social and emotional skills. These social and emotional skills are modeled by creating representations for emotion, personality, and social relationships within their NPCs.

Avatar Arena [35] explores believability in games by creating a system in which simulated negotiations between AI agents are modeled to resemble negotiations between humans [7]. The simulation attempts to produce agents that form rational arguments based on social context and personality. This work focuses on how social relationships impacts the behavior of agents in the simulation.

Hashemian et al. [18] explores the concept of social power in the context of NPC social interaction. By their definition, "Power involves getting individuals to comply with your requests, even if they are reluctant to do so" [18]. The work of Hashemian

et al. focused on accounting for the social power dynamic between NPCs during the NPC's decision-making process.

3.3 Drama Management

The topic of drama management in games focuses on how the narrative of a game is expressed and how that expression changes based on the actions of the human player [39]. While drama management is not strictly limited to NPCs, NPCs do play a large part in expressing the narrative of games.

A Behavior Language [28] (or ABL) is a method for believable agents that draws from both research about social believability and drama management. Their research was realized and showcased through the game *Façade*. Like many modern games, *Façade* features branching outcomes to the story, but the way in which it achieves this branching nature is quite interesting. Mateas and Stern build upon the Hap language [8] [25], which was specifically designed to create believable agents. ABL extends the Hap language to include representation of dramatic beats in the agents. The addition of dramatic beats for agent behavior allowed the characters to more closely influence and adapt according to the drama management of the game.

Chapter 4

PROPUSED SOLUTION

Our goals for this work are threefold. The first is to devise a NPC behavior model that results in differing behavioral outcomes based on initial personality traits. The second is to implement a novel method for procedural animation based on mood. Third, we aim to create a maze simulation environment to analyze our behavior model and procedural animation both visually and statistically.

While others have taken various approaches to leveraging personality and mood to control the behavior of NPCs [26] [24] [14] [15] [9], our approach combines them in a novel way. For our behavioral model, we will create a system in which NPCs will decide their actions based on a combination of factors: their personality traits, their current mood, and other various objective-specific factors. Our behavior simulation uses a greedy approach to decide the actions of NPCs by producing a 'potential value' heuristic to compare the utility of actions. The potential value of an action will be determined by the action's affect on the NPC's mood.

This fluctuating mood is also used to produce facial animation for each NPC. Facial blendshapes that represent discrete moods are placed on our mood graph, and the current mood of the NPC determines which blenshapes are displayed.

The maze simulation environment will include a physical maze for NPCs to navigate. Also, the environment includes a finite set of actions that NPC agents can perform while inside the maze environment. The simulation also includes user interface elements that describe the behaviors and mood of our NPCs in real time. Finally, the

maze simulation environment will be used to collect data about the behaviors of our NPCs in order to analyze the efficacy of our behaviour model.

Chapter 5

DEVELOPMENT TOOLS

Here we will discuss the choices we made for our development environment, including our game engine and the tools that we used to model and animate our NPCs.

5.1 Unity

While the Unreal is also a very popular and robust game engine, we decided to use the Unity game engine [4]. The deciding factor between the engines for us came down to experience; we had much more experience with Unity from previous work, so the choice was clear. Unity also has a wide array of features that helped facilitate the development process. Features like a C# scripting language, character animation, blendshape support, physics simulation, and AI pathfinding were utilized to construct our maze simulation environment.

5.1.1 Unity NavMesh for AI Pathfinding

In Unity, there is this feature for AI pathfinding called the NavMesh. This NavMesh defines a surface, or mesh, that AI agents can traverse. This allows the programmer to create a world, or maze in our case, and then designate which object mesh represents the floor, and then create an navigable area for Unity's AI agents to move around on. This feature was integral to our maze simulation environment. We used it to move and animate our characters, as well as help determine distances between objects in our maze.

5.1.2 Line of Sight Detection

The NPCs in our simulation rely on a line-of-sight detection system for detecting other objects. This was made possible by an asset from the Unity Asset Store called *Sensor Toolkit* [3]. With the *Sensor Toolkit*, we are able to place line-of-sight sensors on each of our NPCs in the maze. Additionally, we assign each NPC and each maze node to be a 'sensor target', which means that these objects are marked as things that the line-of-sight sensor will attempt to detect. Finally, the terrain of our maze had to be designated as geometry that obstructed the line-of-sight detector.

5.1.3 Animating in Unity

The built-in Unity animator component was used to animate our NPCs. In the animator component, the programmer can designate animation states and transitions in the animator state machine. The animation states represent a specific animation (e.g. walking, turning, dancing), while the transitions represent events occurring. A particularly nice feature of the Unity animator is the ability to trigger events from within a script, which we used extensively. The behavior of our NPCs is controlled through a series of scripts, and those scripts are able to easily communicate with the animator component to trigger various animations to play.

5.2 Autodesk Character Generator

A good character model takes a considerable time to create. The Autodesk Character Generator [2] is a robust online character model creation tool that can produce character models in minutes. The models produced include skeleton animation rig-

ging and facial blendshapes, which are both necessary for animating our NPCs. We created all character models in this work with the Autodesk Character Generator.

5.3 Mixamo

Just like character models, character animation is a very time-consuming task. Adobe's Mixamo [1] is an online animation library that allowed us to circumvent the work of creating animations for our models.

Chapter 6

METHODS

In this chapter we will explain our maze simulation environment, our behavioral model, and our method for procedural animation.

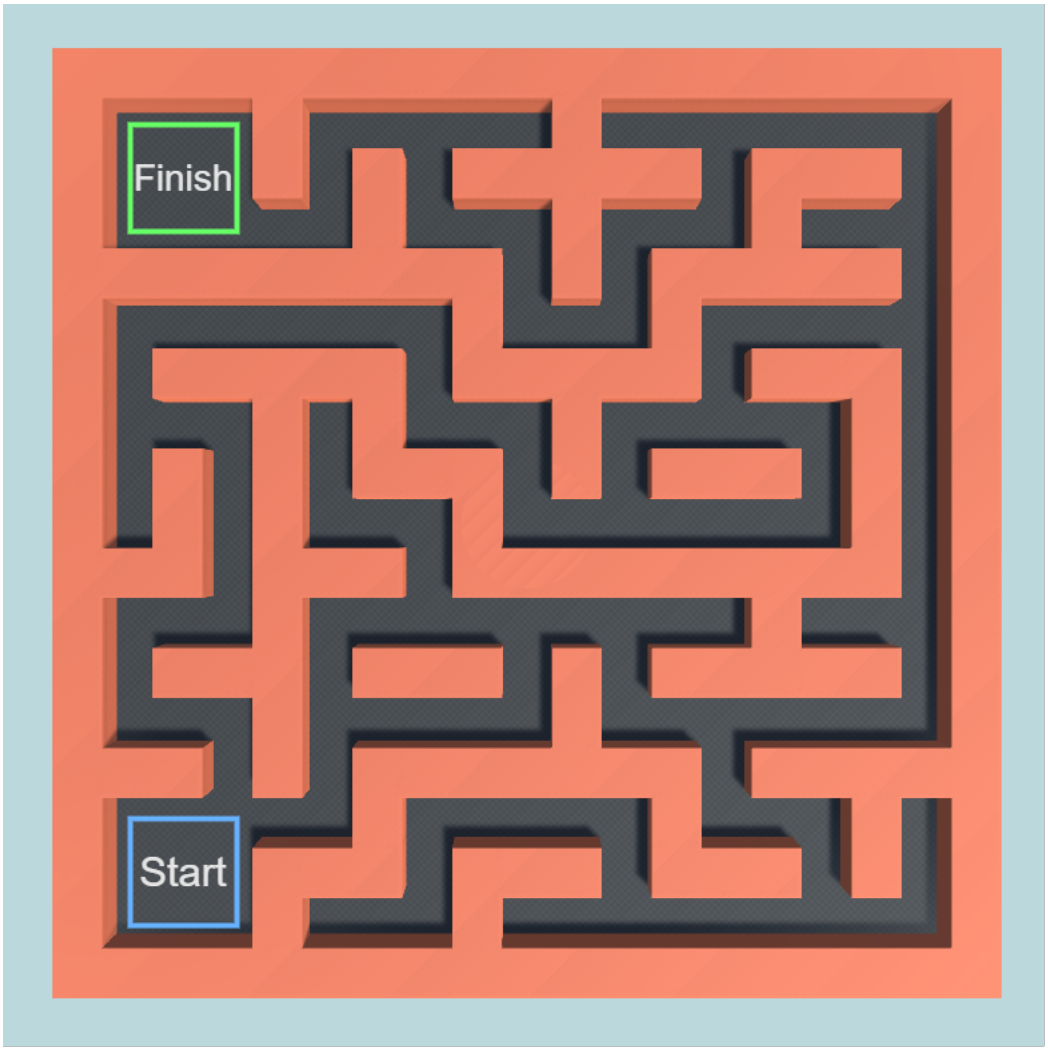

Figure 6.1: The maze used in our simulation. NPCs begin the maze at the starting area, indicated in blue, and attempt to navigate to the finish area, indicated in green.

6.1 The Simulation Environment

We chose the setting of a maze for the sandbox in which our NPCs and their AI would be tested. This environment provides a clear, singular goal for all NPCs, while also affording the ability for NPCs to interact in meaningful ways. The maze, shown in Figures 6.3 and 6.1, is standard maze, where the NPCs will begin at the starting area (see figure 6.1) and must navigate to the end of the maze. At the start of the simulation, NPCs are strangers so they must get acquainted with one another and either learn to work together or attempt to solve the maze on their own. The way each NPC decides to try to solve the maze is dependent on their innate personality as well as their fluctuating mood. By choosing the maze we are making the assertion that this environment is a suitable simulation environment for our NPC AI method, which we believe can be applied across a wide range of video game genres.

Figure 6.2: An NPC in our maze simulation.

6.1.1 The Maze Graph

The way we chose to internally represent the maze was with an connected, undirected graph. In Figure 6.3, notice the blue-green, numbered spheres placed periodically throughout the maze. These spheres are the maze graph's nodes. The nodes were placed strategically at every crossing, corner, and dead-end in the maze. These nodes can be thought of as waypoints – that is, all the possible places an NPC could decide to walk to within our maze.

In order for an NPC to be able to navigate from one node to the next, an adjacency list for each node was also created. In general, the nodes adjacent to a given node, referred to as neighbors, are defined to be the closest node in each of the four cardinal directions where the node is not obstructed by a wall. For example, consider node 10, which is located near the center of the maze. For the sake of explanation, assume that the direction towards the top of Figure 6.3 is oriented with the cardinal direction of North. Node 10 has three neighbors: node 6, node 11, and node 12. Node 6 is the *closest* node to the West; node 11 is the closest *unobstructed* node to the South; node 12 is the closest unobstructed node to the East. Notice that node 10 has no neighbor to the North because there are no unobstructed nodes in that direction. Following this method, an adjacency list for each node was constructed.

Nodes can also be marked as 'visited'. This allows the NPCs to remember where they have been in the maze to better make decisions about where to go next. Section 6.1.3.6 goes into more depth about how NPCs visit nodes.

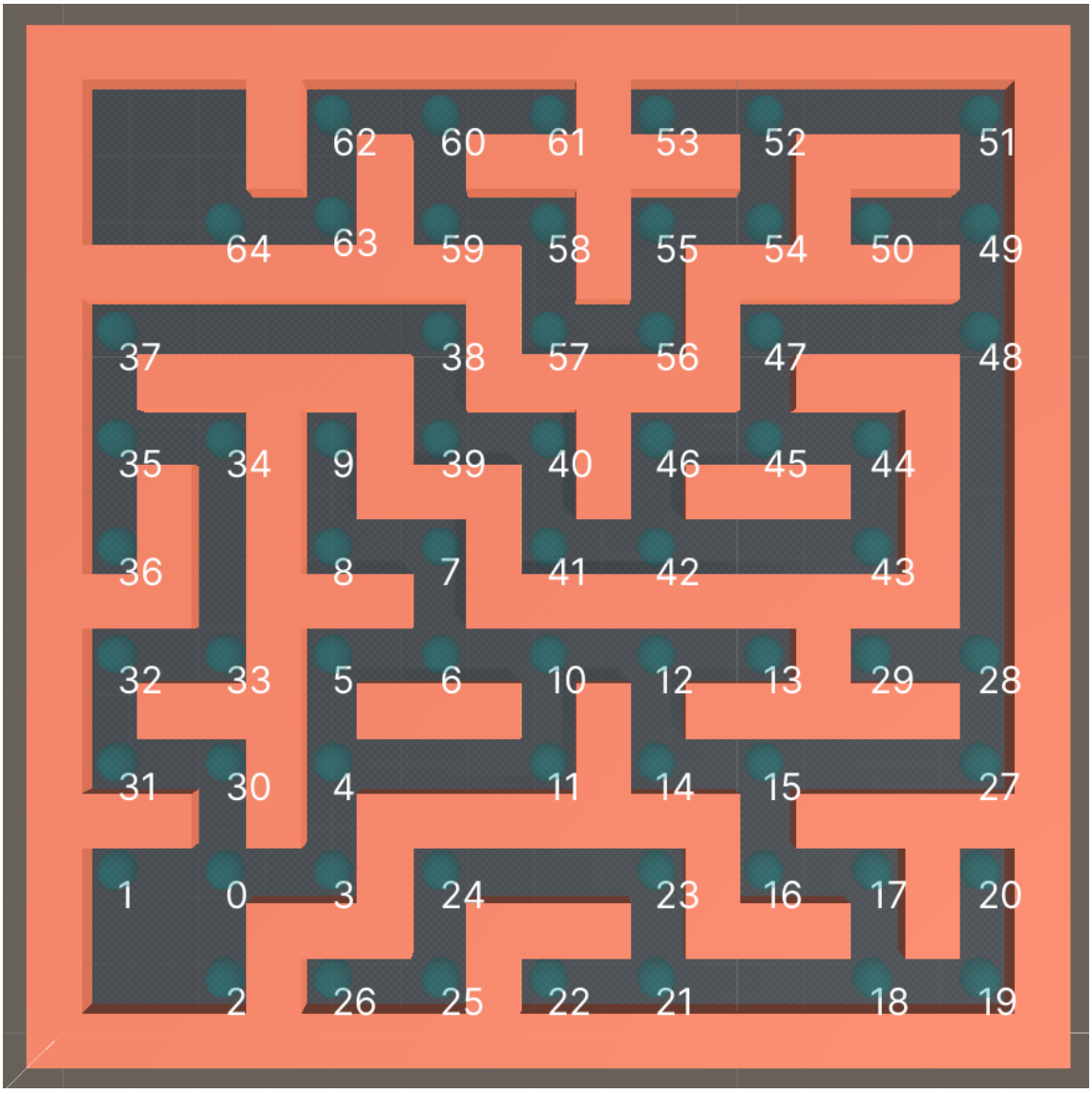

Figure 6.3: Our maze, overlaid with labeled maze nodes. The blue-green numbered spheres show the location of all the maze nodes.

6.1.2 Knowledge and Opinion of Other NPCs

Our maze simulation involves interaction between NPCs, so we felt it necessary for each NPC to contain some form of knowledge and opinion about the other NPCs that they encounter. We model this in two ways: a 'perception' value and a maze graph.

The perception value is a number that ranges from -1 to +1. When an NPC meets another NPC, the other NPC will start with a neutral perception. As the simulation progresses, interactions between the two NPCs – specifically, the positive or negative outcomes of interactions – will then adjust this value. Further, this perception value will be used to determine the value of certain social actions.

The maze graph for other NPCs works in an almost identical fashion as described in section 6.1.1, with the only difference being in how the graph is updated. Given NPCs A and B, NPC A's perceived maze graph for NPC B would include only the nodes that NPC B has shared with NPC A, as well as the nodes that NPC A has shared with NPC B. This may not correctly reflect all the nodes that NPC B has actually visited, but correctly models perceived knowledge of B, given imperfect information. From NPC A's point of view, this representation is correct because NPC A's knowledge of B is limited to their interactions with B. This memory of what has been shared is useful for NPCs when they are deciding what information to share with another NPC.

6.1.3 Actions

Our model of NPC behavior follows a similar approach to the state machine model covered in the section 2.2. The behavior model can be viewed as a form of state machine, where behavior states represent nodes and events represent the transitions between states. In our model, we call our behavior states actions. The transition events generally occur when actions are completed. For a more precise explanation of the transition events, refer to section 6.2.1.

We do consider our method extensible to any behavior model that uses a form of state machine, but we will describe the actions for our particular behavior model in order

to supplement the explanation of our method for state transitions. The following is a description of all the possible actions that an NPC can perform in our simulation.

6.1.3.1 Go To Node Action

Since the nodes represent all the possible places to explore in our maze, we needed an action that lets an NPC explore the maze by navigating to a node. This action consists of an NPC walking towards a maze node, and is considered finished when the NPC collides with the sphere of the targeted maze node.

6.1.3.2 Greet Action

The NPCs all start the maze as strangers, so there must be some action that allows them to get acquainted with one another. This action consists of the NPC turning towards another NPC and waving at them. The action is considered finished when the waving animation is finished playing. This action is special in that, for a given NPC, the 'Follow', 'Share', and 'Wait' actions cannot be performed with another NPC until the given NPC first performs a 'Greet' action.

6.1.3.3 Follow Action

Once acquainted with one another, some NPCs may choose to follow others through the maze in hopes that they will lead them closer to the finish, or just because they don't want to be alone. In this action, an NPC will walk towards and follow another NPC. This action is considered finished when the NPC gets within a certain range of the target, which is about an arm's length away.

6.1.3.4 Share Action

Not only do NPCs store a memory of where they have been in the maze, but they are also able to share that memory with other NPCs in the simulation. In this action, an NPC will follow another NPC until they are within a certain distance to share information about the maze. Once they are close enough, the NPC will begin sharing the information. If they are not able to share all the information before the target NPC moves beyond the share distance, then the NPC will continue to follow their target NPC until they are once again close enough to share. Given NPCs A and B, the information that NPC A can share with NPC B are all the nodes that NPC A has visited. Once the action is complete, NPC B goes into their own maze graph and marks all the nodes that NPC A shared as visited. This action is considered finished after all the information is finished sharing. The way we determine when the information sharing is finished is by use of a timer. The target time limit for the timer scales linearly by the number of nodes being shared.

6.1.3.5 Wait Action

Over the course of the maze, NPCs may develop bonds. To facilitate this, we added the ability for NPCs to wait for each other. NPCs will perform this action when they notice that another NPC is following them or sharing information with them. This action consists of the NPC turning to look at the other NPC and standing still. This action is considered finished once the other NPC is finished following or sharing.

6.1.3.6 Scan Action

Navigating a maze is often disorienting for those trying to solve it, so we created an action that allows NPCs to scan their environment. In this action, an NPC will look at each of the maze nodes and NPCs near them, and may also look at any other NPCs near them. The maze nodes that this action chooses to look at are all the nodes near the NPC that are not obstructed by walls. Other NPCs may be looked at as well if the NPC judges it worth their time to do so. This judgement is based on their perception of other NPCs, as well as their personality traits. The scan action is important because it must be performed before any other action is performed. Only the objects that the NPC chooses to look at may be acted upon. Specifically, a maze node must be looked at before an NPC can walk to it; another NPC must be looked at before the NPC can greet, follow, share, or wait with the other NPC. This action is considered finished once the NPC has looked at everything they have judged worth looking at.

6.2 Behavioral Model

Now that we have defined our simulation environment we can explain how our behavioral model works. This section will cover how personality and mood have been utilized in our method to enhance the NPC behavior model.

6.2.1 Behavior Simulation

Our behavior model follows a similar state machine approach to most NPC behavior models, where actions represent states in the state machine, and events represent transitions. The complete list of possible actions can be found in section 6.1.3. Our

simulation has no explicit representation of an 'event' to trigger transitions. Instead, we transition to a new state when the current action is finished, or when the NPC decides to interrupt their current action. Figure 7.2 shows how the behavior model chooses whether to transition to a new action or to continue with the current action.

Since each action has defined finish conditions, we can check whether those conditions have been met in each update cycle. When an action finishes, the mental state of the NPC is updated to reflect how they feel about completing the action they just finished. This includes applying the change in mood for the action, as well as adjusting the perception of all the NPCs nearby based on perceived value of the completed action. After this, a new action must be chosen. The list of potential actions is populated based on their surroundings, and then the potential value for each action is calculated and compared to the others. The action with the highest potential value is the action that is ultimately chosen by the NPC. If there are ties, then ties are broken by choosing the action that the NPC predicts will take less time to complete. If there are still ties then the NPC simply picks a random action among the ones tied for best. It is important to note that actions rarely have the exact same predicted time to complete, so this random choice rarely occurs.

The NPCs don't always finish every action that they choose to do because they can also interrupt their current action. A repopulation timer is always running, and when that timer reaches its limit the NPC populates their list of potential actions from their surroundings and chooses whether to switch to a new action based on potential value and projected time to complete.

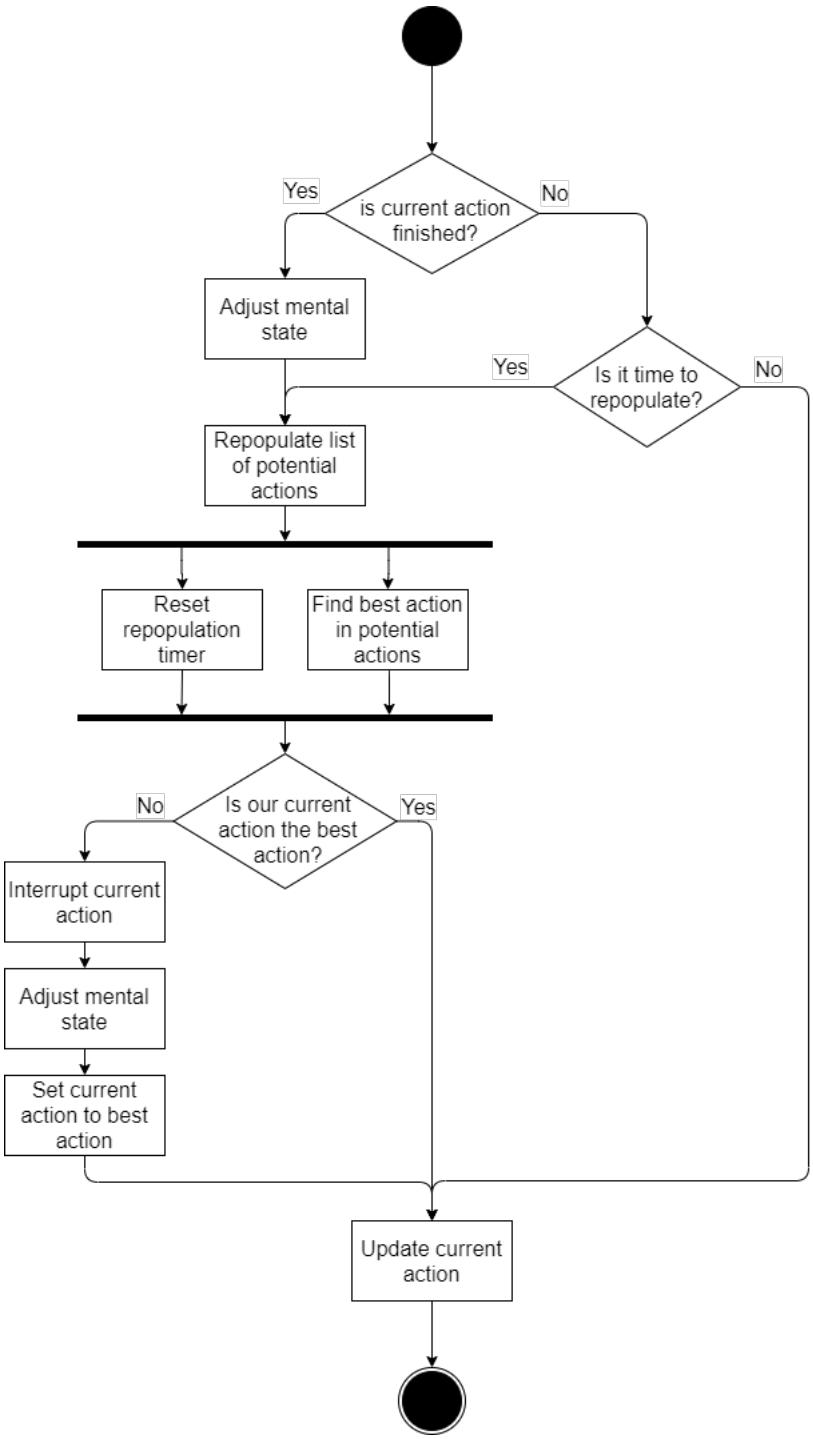

Figure 6.4: The flowchart describing a single update cycle of the behavior controller for our NPCs. Behaviour is determined using a heuristic to perform a greedy selection of actions.

6.2.2 Populating the List of Possible Actions

When an action is completed or interrupted a list of possible actions must be populated so that the NPC may choose a new action. The way this list is populated is determined by what is in sight of the NPC. The NPCs can only act on what they can see. The list of potential actions is populated by taking everything in sight (and everything seen recently, which is defined as things that have been seen at most 15 seconds ago) and adding all the possible actions associated with the object in sight. Additionally, it is important to note that the object must first be marked as looked at before its associated actions may be added to the list. If an NPC is in sight, then greet, follow, and share actions may be added to the list. If a maze node is in view, then a go to node action is added to the list.

The scan action is a special case. The scan action is added to the list if at least one new object (maze node or NPC) has been recently detected. The scan action will look at all the maze nodes near the NPC and possibly some of the other NPCs nearby, and then once the objects have been marked as looked at then they may be added to the list of potential actions.

6.2.3 Mental State

The mental state is home to the personality, mood (desired and current), and knowledge about other NPCs. When an action is finished, the mental state must be updated to apply the change in mood for the action, as well as to reflect any changes about the knowledge of other NPCs (perception and maze graph).

6.2.4 Personality Representation

One component of our NPC behavior model that helps inform transitions between behavior states is our representation of personality for each NPC. In our personality representation we follow a similar approach to Li and MacDonnell [24] in that each NPC receives a numerical score for each of the five personality traits (openness, conscientiousness, extraversion, agreeableness, neuroticism). The numerical scores range from 0 to 1, where 0 indicates that the NPC does not exhibit the behaviors associated with this trait at all, and 1 indicates that the NPC fully exhibits the behaviors associated with the given trait. The personality scores for each NPC can be set before each run of our simulation and do not change once the simulation begins.

6.2.5 The Mood Graph

We chose to include a changeable mood to our behavior model. The way we model mood is very similar to the circumplex model of affect [37]. Figure 6.5 shows a visual representation of the mood graph we used in our behavior model. Our mood graph has two axes: energy (which correlates to Russell's 'arousal' axis) and pleasantness. The values along each axis are restricted to lie within -1 and +1. When an NPC completes an action, their mood is updated to reflect how the completion of the action makes them feel. The mood graph is also used to animate the facial expressions of our NPCs. For more on how this is done, refer to section 6.3.

6.2.5.1 Desired Mood

The desired mood for an NPC is a position on the mood graph that is pre-calculated when the simulation begins, and is based on the NPC's personality traits. See Figure

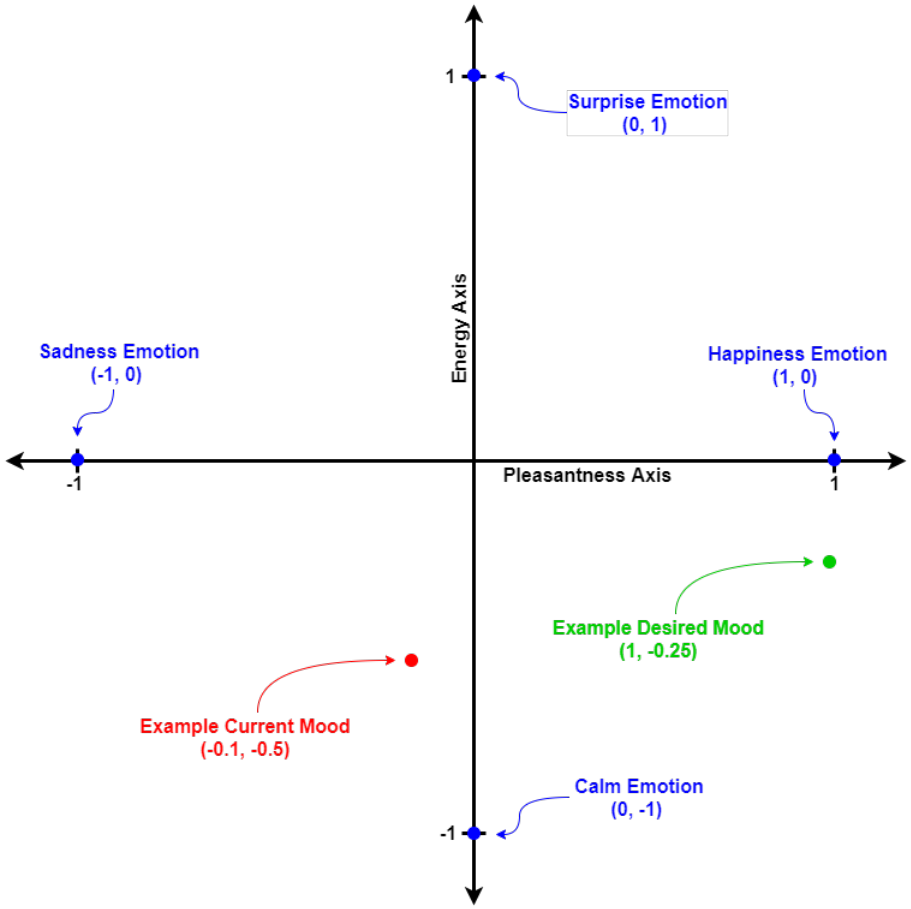

Figure 6.5: Our graph of emotions, based on Russell's circumplex model of affect [37]. Like Russell's graph, we label emotions as positions on the graph (marked in blue). For simplicity, we have chosen four basic emotions that align well with the poles of our graph: happiness at (1, 0), calm at (0, -1), sadness at (-1, 0), and surprise at (0, 1). Further, we have labeled examples of current mood (in red) and desired mood (in green).

6.5 for an example position for desired mood. Since the desired mood is determined by the NPC's personality, every NPC has their own distinct desired mood. While the current mood of an NPC fluctuates as the simulation progresses, the desired mood stays constant throughout.

The desired mood consists of two values: an energy value and a pleasantness value. We chose to hold the desired pleasantness constant across all NPCs, assigning it a value of 1, which is maximum pleasantness. By assigning the desired pleasantness in

this way we are able to model how humans tend to generally seek happiness in life. Additionally, this allows us to easily compare actions of the same type. For instance, walking towards a dead-end in the maze, or walking to an undiscovered area. If the goal is to solve the maze, then the obvious better choice is to walk to an undiscovered area. Since all NPCs desire a pleasantness of +1, we can produce a higher, positive change in pleasantness for walking towards an undiscovered area and be sure that this will increase the value of this action (see section 6.2.8 for exactly how potential value is calculated).

Conversely, the value for desired energy is completely dependent on personality traits. The traits that we determined would affect desired mood are conscientiousness, extraversion, and neuroticism. The following formula was used to calculate desired energy:

$$DesiredEnergy = 0.5 * \epsilon + 0.5 * P_N * \omega - 0.5 * P_C * \omega \tag{6.1}$$

Where,

$$\epsilon = (P_E * 2) - 1 \tag{6.2}$$

$$\omega = \frac{\epsilon}{|\epsilon|} \tag{6.3}$$

The symbols P_C, P_E, and P_N correspond to the NPC's personality traits of conscientiousness, extraversion, and neuroticism, respectively. ϵ represents extraversion (which ranges from 0 to 1) mapped to a value between -1 and +1. ω is used to capture the sign of ϵ.

Recall that the energy axis on the mood graph ranges from -1 to +1, so the equation above was designed to produce values within this range. The first term in the equation, $0.5 * \epsilon$ gives a number that ranges from -0.5 to +0.5. The second term,

$0.5 * P_N * \omega$, increases the absolute value of desired energy based on the magnitude of neuroticism, while the third term, $-0.5 * P_C * \omega$ decreases the absolute value of desired energy.

6.2.6 Calculating E_{Factor} and P_{Factor} for Each Action

Before the potential value of an action can be calculated, an action must produce two values: E_{Factor} and P_{Factor}. These values are numbers that range from 0 to +1, and represent the effect of a given action on an NPC's mood. After the values are produced, E_{Factor} is used to affect a change in energy, and P_{Factor} is used to affect a change in pleasantness. When $0 <= E_{Factor} < 0.5$, a negative change in energy is produced; when $E_{Factor} = 0.5$, no change in energy is produced; when $0.5 < E_{Factor} <= 1$, a negative change in energy is produced. The P_{Factor} works similarly, except it affects a change in pleasantness. Equations 6.15 and 6.16 show exactly how E_{Factor} and P_{Factor} are used to produce a change in mood. The following subsections will describe how E_{Factor} and P_{Factor} are calculated for each action.

6.2.6.1 Calculating E_{Factor} and P_{Factor} for the Go to Node Action

The following equations describe how E_{Factor} and P_{Factor} are calculated for the go to node action:

$$E_{Factor} = lerp(0.5, 0, P_E) \tag{6.4}$$

$$P_{Factor} = 0.25 * NodeInterest + 0.75 * NodeInterest * P_C \tag{6.5}$$

where,

$$NodeInterest = \frac{1}{\gamma} \tag{6.6}$$

The E_{Factor} equation above uses the 'lerp' function, or linear interpolation, which is defined as $lerp(a, b, x) = a + (b - a) * x$, and is used to blend between two values. The P_E and P_C symbols represent the NPC's extraversion and conscientiousness, respectively. the γ symbol represents the distance from an unexplored node. It is important to note that the distance from an unexplored node is guaranteed to be at least 1, so that we do not divide by zero. This particular E_{Factor} equation (equation 6.4) produces values that range from 0 to 0.5, and depends on the NPC's value for extraversion. The P_{Factor} (equation 6.5) consists of a couple terms, so we will explore each one. The first term, $0.25 * NodeInterest$, is added to this equation to make sure that P_{Factor} can output a nonzero value, even if the NPC's conscientiousness is zero. This is useful for comparing go to node actions for NPCs that have a conscientiousness score of 0. The second term, $0.75 * NodeInterest * P_C$, is very similar to the first, but is multiplied by conscientiousness. This ensures that the NPC must have a sufficiently high conscientiousness score in order to have a positive change in pleasantness.

6.2.6.2 Calculating E_{Factor} and P_{Factor} for the Follow Action

The following equations describe how E_{Factor} and P_{Factor} are calculated for the follow action:

$$E_{Factor} = 0.75 \tag{6.7}$$

$$P_{Factor} = 0.5 * perception + 0.5 * P_E \tag{6.8}$$

In equation 6.8, the symbols *perception* and P_E correspond to the perception of the NPC that is to be followed (see equation 6.22 for a precise definition of perception) and the NPC's extraversion, respectively. The E_{Factor} – equation 6.7 – is hard-coded to be 0.75 to make it more likely for highly extroverted NPCs to choose this action,

since high extroversion produces a higher desired energy. Equation 6.8 is designed to produce a P_{Factor} that is equally influenced by both perception and extroversion.

6.2.6.3 Calculating E_{Factor} and P_{Factor} for the Share Action

The following equations describe how E_{Factor} and P_{Factor} are calculated for the share action:

$$E_{Factor} = 0.75 \tag{6.9}$$

$$P_{Factor} = 0.125 * P_A + 0.125 * P_E + 0.75 * \beta \tag{6.10}$$

where,

$$\beta = \frac{numNodesToShare}{numVisitedNodes} \tag{6.11}$$

In equation 6.10, the symbols P_A and P_E represent the NPC's agreeableness and extraversion, respectively. In equation 6.11, the symbols β, $numNodesToShare$, and $numVisitedNodes$ represent the usefulness of the information being shared, the number of nodes being shared with this action, and the number of total nodes that the sharer NPC has visited. Similar to equation 6.7, the share action E_{Factor} (equation 6.9) is hard-coded to be 0.75 to make it more likely for highly extroverted NPCs to choose this action, since high extroversion produces a higher desired energy. For the P_{Factor} calculation (equation 6.10), agreeableness and extroversion are given relatively small weight coefficients, while the usefulness value, β, is given a much stronger weight coefficient. This is to make sure that NPCs will only share information when the information includes a significant number of nodes.

6.2.6.4 Calculating E_{Factor} and P_{Factor} for the Wait Action

The following equations describe how E_{Factor} and P_{Factor} are calculated for the wait action:

$$E_{Factor} = P_E \tag{6.12}$$

$$P_{Factor} = 0.5 * P_E + 0.5 * P_E * perception \tag{6.13}$$

In equations 6.12 and 6.13 above, P_E represents the extroversion of the NPC. In equation 6.13, *perception* represents the perception of the current NPC about the NPC that they are considering waiting for. The E_{Factor} is set to the NPC's value for extroversion so that the change in energy produced by this E_{Factor} will be desirable to any personality, since the polarity of desired energy is dependent on the value of extroversion (see equation 6.1). Equation 6.13 calculates P_{Factor} for the wait action, and consists of two terms. While both terms multiply by P_E, the first term, $0.5 * P_E$ ensures that extroverted NPCs will get a reasonably high P_{Factor}, despite the perception of the other NPC. The second term, $0.5 * P_E * perception$, is included to deter NPCs from waiting for other NPCs that they do not like.

6.2.6.5 Calculating E_{Factor} and P_{Factor} for the Scan Action

The scan action behaves a bit differently than other actions. This action is designed to produce an artificially high potential value so that NPCs choose to scan the environment, since this is absolutely necessary in a maze environment. When a scan action is complete, it has no effect on mood – that is, the E_{Factor} and P_{Factor} are both set to 0.5.

6.2.7 Calculating a Change in Mood

Every action produces a change in mood. Each action first computes a E_{Factor} and P_{Factor}, and then the E_{Factor} and P_{Factor} are used to determine a change in mood. A change in mood is expressed by a change in energy and pleasantness. The following equations provide a precise definition for how these values are calculated:

$$\Delta_{Mood} = (\Delta_E, \Delta_P) \tag{6.14}$$

where:

$$\Delta_E = lerp(-\tau, \tau, E_{Factor}) * (1 - |\Psi_E|) \tag{6.15}$$

$$\Delta_P = lerp(-\tau, \tau, P_{Factor} + \lambda) * (1 - |\Psi_P|) \tag{6.16}$$

$$\lambda = lerp(0, -0.25, P_N) \tag{6.17}$$

$$\tau = lerp(0.1, 0.5, P_N) \tag{6.18}$$

The symbols Δ_{Mood}, Δ_E, and Δ_P represent the change in mood, the change in energy, and the change in pleasantness, respectively. The symbols Ψ_E and Ψ_P represent the current mood's values for energy and pleasantness, respectively. Recall that the energy and pleasantness values for mood range from -1 to +1. The E_{Factor} and P_{Factor} are values that are pre-calculated by each action, and are guaranteed to be bounded between 0 and 1. Section 6.2.6 describes exactly how E_{Factor} and P_{Factor} are calculated for each action. P_N represents the NPC's personality trait of neuroticism, which ranges from 0 to 1. Many of the equations above use the 'lerp' function, or linear interpolation, which is defined as $lerp(a, b, x) = a + (b - a) * x$, and is used to blend between two values.

τ represents the bounds for possible delta values. This value itself is bounded between 0.1 and 0.5, which are arbitrary, but have been found to work sufficiently well by producing a change in mood that is large enough to be noticed, but not so large that mood swings drastically. The Δ_E and Δ_P values are bounded by this τ value – specifically, $-\tau$ and $+\tau$. This ensures that both delta values lie within a symmetric range, and that the magnitude of the range is directly correlated with the magnitude of neuroticism. This produces larger changes in mood for NPCs who are more neurotic, which emulates real life data [45]. A neurotic skew, λ, is also applied to Δ_P, which mimics a correlation between neuroticism and less pleasant emotional reactions to events [21].

In equation 6.15, the result of the lerp function is multiplied by $(1-|\Psi_E|)$, which alters the the value for Δ_E so that the magnitude decreases as the current approaches the boundaries of the energy axis on the mood graph (both axes on the mood graph are bounded from -1 to +1). In equation 6.16, the result of the lerp function is multiplied by $(1 - |\Psi_P|)$ to achieve a similar result for the pleasantness axis. This achieves a 'sticky' effect as the current mood approaches the edges of any axis.

6.2.8 Potential Value

Potential value is our heuristic for comparing the utility of different actions in our simulation. Just like in life, some actions are better than others, and some actions may have positive or negative effects. This effect of real-life actions is simulated with our potential value calculation. Each action produces a potential change in mood, and that potential change is mood is used to calculate our potential value heuristic. This heuristic is calculated based on the distance between desired mood and the potential position of the current mood if an NPC were to complete the given action. The following equation provides a more precise definition:

$$PotentialValue = \frac{(D_C - D_P)}{\sqrt{8}} \tag{6.19}$$

where,

$$D_P = |P_D - (\Delta_{Mood} + P_C)| \tag{6.20}$$

$$D_C = |P_D - P_C| \tag{6.21}$$

The symbols D_P and D_C represent the distance from a potential mood to the desired mood, and the distance from the current mood to the desired mood, respectively. The difference in distances is divided by the maximum possible distance between two points on our graph, which evaluates to $\sqrt{8}$. We do this to ensure that the potential value never exceeds the bounds of -1 to +1. Bounding the potential value in this way is useful for calculating change in perception (see section 6.2.9). The symbols P_D, P_C, and Δ_{POT} represent the position of the desired mood, the position of the current mood, and the potential change in mood, respectively. The potential change in mood, Δ_{Mood}, is calculated for each potential action (see equation 6.14 for how this is calculated).

6.2.9 Calculating Change in Perception

Recall from section 6.1.2 that perception of others is represented with a single value that ranges from -1 to +1. Given NPCs A and B, NPC A's perception of B changes based on the potential value of an action completed in the vicinity of B, and the neuroticism of A. The following is the precise equation for calculating change in perception:

$$Perception = Perception + \sigma * (PotentialValue + \phi) \tag{6.22}$$

where:

$$\sigma = -\frac{(1 - |2 * Perception - 1|)}{2} \tag{6.23}$$

$$\phi = -lerp(0, |PotentialValue|, P_N * P_A) \tag{6.24}$$

The symbols σ and ϕ represent a distance from the boundaries (perception is bounded between -1 and +1) and a skewed potential value, respectively. This distance from the boundaries value, σ, alters the change in perception so that the magnitude of the change in perception decreases as the perception value approaches the boundary values. This 'sticky' effect models a history of experience between NPCs. The ϕ value represents a narcissist skew, which is determined by P_N and P_A, the NPC's traits for neuroticism and agreeableness. As both P_N and P_A increase, an NPC will have worse perceptions of the other NPCs because ϕ is guaranteed to always be negative. If P_N and P_A are both 1, then the NPC is not able to increase perception of others – instead, perception will only stay the same or decrease.

Figure 6.6: Facial animation blendshapes were mapped to moods, and then the moods were mapped to positions on our mood graph. This shows the resulting facial animations as the current mood moves to different positions on the mood graph. From left to right: (1) medium energy and medium pleasantness (control), (2) medium energy and low pleasantness, (3) medium energy and high pleasantness, (4) low energy and medium pleasantness, (5) low energy and low pleasantness, (6) low energy and high pleasantness, (7) high energy and medium pleasantness, (8) high energy and low pleasantness, (9) high energy and high pleasantness.

6.3 Procedural Animation with Mood and Blendshapes

The mood graph is used in two ways. As already discussed, the potential value for actions in our behavior model is determined by current and desired mood. Another function of our mood graph is to drive facial animation. The mood graph shown in Figure 6.5 shows the four emotion blendshape targets (in blue) and their corresponding positions on the graph. The facial animation of each NPC is determined by where the current mood lies on our mood graph. If the current mood is in the top-right quadrant of the graph then facial blendshapes are blended between surprise and happiness blend targets. If the current mood is in the bottom-right quadrant of the graph then facial blendshapes are blended between calm and happiness blend targets. If the current mood is in the bottom-left quadrant of the graph then facial blendshapes are blended between calm and sadness blend targets. If the current mood is in the top-left quadrant of the graph then facial blendshapes are blended between surprise and sadness blend targets. Figure 6.6 shows the facial animations produced for varying positions of mood on the mood graph.

Chapter 7

RESULTS

Here we will analyze the results of our behaviour model, the maze simulation environment, and the mood-based procedural animation.

7.1 Data Collection

Data was collected over the course of several runs of the maze simulation. Each run contained 10 NPCs with randomized personality traits, and the simulation run was considered finished only when all of the NPCs had reached the finish node in the maze. We collected data for each NPC about their personality traits, which actions they performed, and the average distance from their current mood to their desired mood. Tables 7.1 and 7.2 show the data we have collected. We ran 100 simulations consisting of 10 NPCs per simulation, resulting in 1000 data points.

To measure the correlation of traits and actions (see Table 7.1), we recorded the fraction of the number of a specific type of chosen action over the total number of chosen actions. It is important to make the distinction of *chosen* actions, since actions may be interrupted, which means that some portion of chosen actions are never completed. The four actions we decided to measure were: the go to node action, the follow action, the share action, and the wait action.

We chose to omit the look, scan, and greet actions. The look and scan actions are actions that are given an artificially high value (regardless of personality traits) so that the NPCs will scan often. This trait-agnostic value for look and scan actions

means that no correlation between traits and these actions can be drawn. As for greet actions, we chose to omit them because greeting only occurs at most nine times and NPCs will typically complete over 100 actions in a single simulation.

The data for Table 7.2 was collected by taking the average distance from the desired mood to the current mood, each frame. If the NPC reached the end of the maze, then we stopped adding samples to the average because the NPC's mood does not change after it reaches the finish. Since each NPC finishes at different times, we did not want to continue to sample the mood of an NPC after it finished.

Table 7.3 includes data about the distance to the closest NPC. This data was collected by iterating over all the NPCs in the scene and determining which NPCs are closest to each other, every frame. After the simulation is over, the average of the distance to the closest NPC, per NPC in our simulation, was calculated by dividing by the total number of frames. It is important to note that the measured distance between NPCs takes into account paths around the walls of our maze.

After data was collected, we calculated the Pearson product-moment correlation coefficient, r. This coefficient ranges from -1 to +1 and is used to measure the linear relationship between two variables. A positive value for r indicates that the datasets are directly correlated, while a negative value indicates that the data is indirectly correlated. The magnitude of r represents how closely the data is correlated, with $|r| = 0$ indicating that there is no correlation, and $|r| = 1$ indicating that the data is perfectly correlated. The r value can be classified into 3 categories based on strength of association: strong, medium, and weak. For strongly correlated data $0.5 < |r| \leq 1$, for medium correlated data $0.3 < |r| \leq 0.5$, and for weakly correlated data $0.1 < |r| \leq 0.3$. The Pearson product-moment coefficient was calculated for both tables 7.1 and 7.2.

7.2 Analysis of the Behavior Model

Recall that our goal was to devise a NPC behavior model that results in differing behavioral outcomes based on the initial personality traits. The maze simulation environment was created to test this behavior model. In the context of our maze, differing behavioral outcomes are represented by the actions that a given NPC chooses to perform. Put simply, the desired results would show that the actions of our NPCs change when their personality traits change.

Trait Versus Action **r** Correlation

trait	***r** Correlation*			
	Go To Node	Follow	Share	Wait
Openness	-0.0645	-0.0178	0.1245	0.0048
Conscientiousness	0.0146	-0.0144	0.1805	0.0159
Extraversion	-0.7166	0.7970	0.1477	0.5224
Agreeableness	-0.0427	0.0061	0.0694	-0.0488
Neuroticism	0.0702	-0.0071	0.0709	0.0864

Table 7.1: The correlation between each personality trait and the go to node, follow, share, and wait actions. The correlation value, r, is the Pearson product-moment correlation coefficient.

Table 7.1 shows the r value for each combination of personality trait and our set of actions. This allows us to analyze the effects of traits on different actions. We will discuss each action and its corresponding correlations for each trait.

7.2.1 Go To Node vs. Trait Correlation Analysis

The go to node action shows virtually no correlation for openness, conscientiousness, agreeableness, and neuroticism. For openness, agreeableness, and neuroticism, this makes sense because those are not factors that influence the calculation of the go to node action's E_{Factor} and P_{Factor} in equations 6.4 and 6.5. However, conscientiousness

does affect the go to node action's P_{Factor} in equation 6.5, so at first it might seem that this does not make sense. Despite this, this correlation can be explained by the way the actions were populated into the list of potential actions. When an NPC is near other NPCs, they have the option to follow or share or wait for others, but if they are alone then their only options for actions are to scan and to go to a node. This does not change, no matter the NPC's personality.

The go to node action is found to have a negative, strong correlation with extraversion. This can be attributed to an indirect effect from the follow, share, and wait actions, since the calculation of P_{Factor} for the follow, share, and wait actions increases as extraversion increases. Thus, the NPCs are more likely to choose some other action other than a go to node action if they have a high extraversion rating.

7.2.2 Follow vs. Trait Correlation Analysis

The follow action shows nearly no correlation for openness, conscientiousness, agreeableness, and neuroticism. This is consistent with our equation 6.8, since it does not use these traits to calculate E_{Factor} and P_{Factor} in equations 6.7 and 6.8.

Extraversion has a strong positive correlation to the follow action. This is consistent with our equation 6.8, which results in a higher value for P_{Factor} as extraversion increases. This is also consistent with the E_{Factor} equation for the follow action, equation 6.7, and the equation for calculating desired energy, equation 6.1. Since NPCs with high extraversion will have positive desired energy, then it would make sense for an NPC with high extraversion to choose to follow due to the E_{Factor} for a follow action being relatively high.

7.2.3 Share vs. Trait Correlation Analysis

Almost no correlation is found between the share action and the agreeableness and neuroticism traits. For neuroticism, this is consistent with equations 6.9 and 6.10, since they does not use neuroticism to calculate E_{Factor} and P_{Factor}. As for agreeableness, it does contribute to the calculation of P_{Factor} in equation 6.10, but it only contributes to $\frac{1}{8}^{th}$ of P_{Factor}. This is must be the reason for the low correlation.

Interestingly, the share action shows a weak positive correlation to conscientiousness. Equations 6.9 and 6.10 show that conscientiousness is not used in the calculation of E_{Factor} or P_{Factor}. Nevertheless, there is a positive correlation. The positive correlation here must be due, indirectly, to β, the usefulness of the information being shared, from equation 6.10. This usefulness value increases with the number of nodes being shared. Since conscientious NPCs are more likely to venture out and explore on their own, if they eventually meet back up with other NPCs then this usefulness value will be very high. This high usefulness value will raise the potential value of the share action, resulting in a higher likelihood of this action being chosen.

Lastly, the share action shows a weak positive correlation to extraversion. Just like agreeableness, equation 6.10 show that extraversion only accounts for $\frac{1}{8}^{th}$ of P_{Factor}. Since agreeableness showed nearly no correlation to the share action, it seems like extraversion should follow suit. However, it does not match the correlation for agreeableness. This may be attributed to the fact that highly extroverted NPCs are likely to often choose to follow and to wait, so they are often near other NPCs. Since they are near other NPCs, they will be given the option to share information with others. If they were not near other NPCs, then they would not even have the option to share information.

7.2.4 Wait vs. Trait Correlation Analysis

The wait action shows basically no correlation for the openness, conscientiousness, agreeableness, and neuroticism traits. This is consistent with equations 6.12 and 6.13, since these trait are not used to calculate E_{Factor} or P_{Factor} for the wait action.

The extraversion trait shows a strong positive correlation to the wait action. This is consistent with equation 6.13, since extraversion is used to calculate P_{Factor} for the wait action. This is also consistent with equations 6.12 and 6.1, since highly extroverted NPCs are likely to have positive desired energy, and since the E_{Factor} for the wait action is hard-coded to give a positive change in energy.

7.2.5 Trait vs. Distance to Desired Mood Analysis

In analyzing Table 7.2, it is important to note that a negative correlation shows that distance from the current mood to the desired mood gets smaller as the given trait gets higher. For positive correlation, that means that distance from the current mood to the desired mood gets bigger as the given trait gets higher. Recall that, ultimately, the goal of an NPC when choosing what action to perform is to get closer to their desired mood.

Trait Versus Distance to Desired Mood **r** Correlation

trait	***r** Correlation*
Openness	-0.0816
Conscientiousness	-0.6751
Extraversion	-0.2678
Agreeableness	-0.0461
Neuroticism	0.2857

Table 7.2: The correlation between each personality trait and the average distance from the NPC's current to desired moods. The correlation value, r, is the Pearson product-moment correlation coefficient.

Table 7.2 shows a strong negative correlation for conscientiousness, a weak negative correlation for extraversion, a weak positive correlation to neuroticism, and nearly no correlation for openness and agreeableness.

The strong negative correlation to conscientiousness shows that highly conscientious NPCs were, on average, closer to their desired mood at any given time. This is likely due to the combination of how we populated the list of potential actions and how we calculated the P_{Factor} for the go to node action. Whenever an NPC is deciding their next action, the list of possible actions will always contain a go to node action, which is due to the way we have distributed our nodes across the maze. Since the P_{Factor} equation for the go to node is strongly dependent on the conscientiousness of the NPC, it makes sense that highly conscientious NPCs would have a higher pleasantness in their mood.

The weak negative correlation to extraversion shows that highly extroverted NPCs were, on average, closer to their desired mood at any given time. Like the correlation for conscientiousness, this is likely due to the combination of how we populated the list of potential actions and how we calculated the P_{Factor} for the follow, share, and wait actions. One might expect the correlation for extroversion to be stronger, like the correlation to conscientiousness, since the P_{Factor} calculations for the follow, share, and wait actions are analogous to the P_{Factor} calculation for the go to node action (if we substitute conscientiousness with extraversion). However, this incongruity can be attributed to the way we populate the list of potential actions. Although highly extroverted NPCs tend to stay close to other NPCs, they do sometimes become separated from others. When a highly extroverted NPC is alone, they are forced to navigate through the maze using only go to node actions. Since the go to node action is designed to make highly conscientious NPCs happy (and not necessarily highly extroverted NPCs), this is why we see a smaller correlation.

The weak positive correlation to neuroticism shows that highly neurotic NPCs were, on average, further away from their desired mood at any given time. This is to be expected, since the potential value equation, equation 6.19, includes a neurotic skew (λ). The λ value decreases the change in pleasantness for any given action by up to 0.25, which ultimately results in an overall lower pleasantness for highly neurotic NPCs. Since the desired pleasantness for every NPC is hard-coded to be 1 (the maximum value for pleasantness), and since highly neurotic NPCs have a lower pleasantness in their mood, it follows that highly neurotic NPCs should be further from their desired mood, on average.

7.3 Summarizing Behaviour Model Analysis

As we have mentioned several times, our goal was to devise a NPC behavior model that results in differing behavioral outcomes based on the initial personality traits. An analysis of Tables 7.1 and 7.2 shows that we have succeeded in producing a behavior model that results in differing behavioral outcomes based on personality traits. In our maze simulation, the differing behavioural outcomes is represented by the actions that an NPC decides to perform. Table 7.1 shows that we measure significant correlations between personality traits and chosen actions. These correlations show that personality traits can be used to affect the behavior of NPCs.

It is unfortunate that we did not observe a significant correlation between the traits for openness and agreeableness and the average distance to desired mood in Table 7.2. Ideally, each personality trait would show a some type of significant correlation to distance to desired mood. This would show that each personality trait has been sufficiently integrated into the simulation environment, which would indicate that we have maximized the opportunity achieve differing outcomes based on the variation of

personality. While we have not *maximized* the differing outcomes, we have at least shown that a variation of behavior is possible. Given our heavy use of conscientiousness and extraversion in comparison to openness and agreeableness in our model, we assert that this failure to maximize differing outcomes is due to the limited types of actions in our maze simulation environment.

7.4 The Maze Simulation

Our maze simulation environment was created with the sole purpose of testing our NPC behavior model. Figure 7.1 shows what the maze simulation looks like while it is running. For the purposes of live analysis and demonstration, we included several user interface features that explain what is happening in the scene in real time.

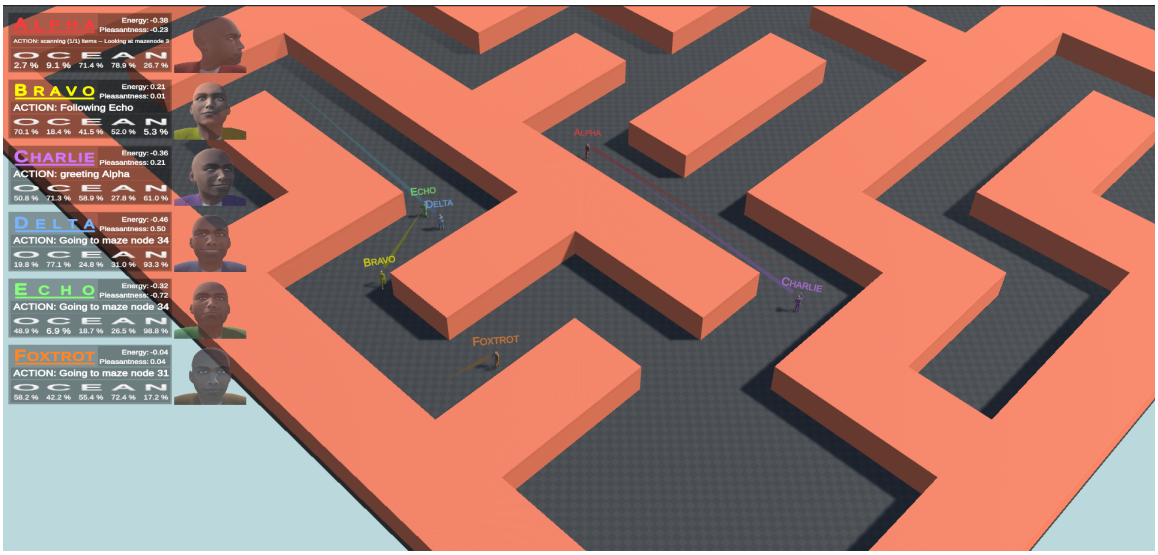

Figure 7.1: A screen capture of the maze simulation, including the user interface elements that help the viewer understand what is happening.

On the left side of the screen, a list of information cards for each NPC displays relevant information. The information on the cards includes the NPC's name, their current action, their personality traits, and and their current mood. To the right of

each information card is a superimposed camera view of the NPC's face, which is used to analyze facial animation.

In addition to the screen-space user interface items, we also included world-space interface. The name of each NPC hovers above each NPC's head as they move around the maze, allowing observers to easily recognize who is who. The other world-space user interface feature we included was the line-of-sight line. This line protrudes from just in front of the NPC's face, and extends towards whatever they are looking at. This gives observers a good idea of what an NPC may be thinking or doing, without the need to read what action they are performing.

7.4.1 The Effectiveness of the Maze as a Behavior Model Testing Environment

As mentioned in section 7.3, the maze as a simulation environment is not without faults. The maze environment is a quite limited testing environment compared to the settings of modern role-playing games. However, we believe the maze environment served our purposes sufficiently, especially when considering the development load. By restricting our environment to a maze, we were able to create a relatively simple simulation environment for our behavior model. In some ways, this simplicity has been beneficial because we did not have to spend much time creating the environment that our NPCs lived in.

While it was not the focus of our work, it is encouraging to find that the design of the actions in our maze simulation environment promoted the formation of groups. Table 7.3 shows a weak negative correlation between the extroversion personality trait and the distance

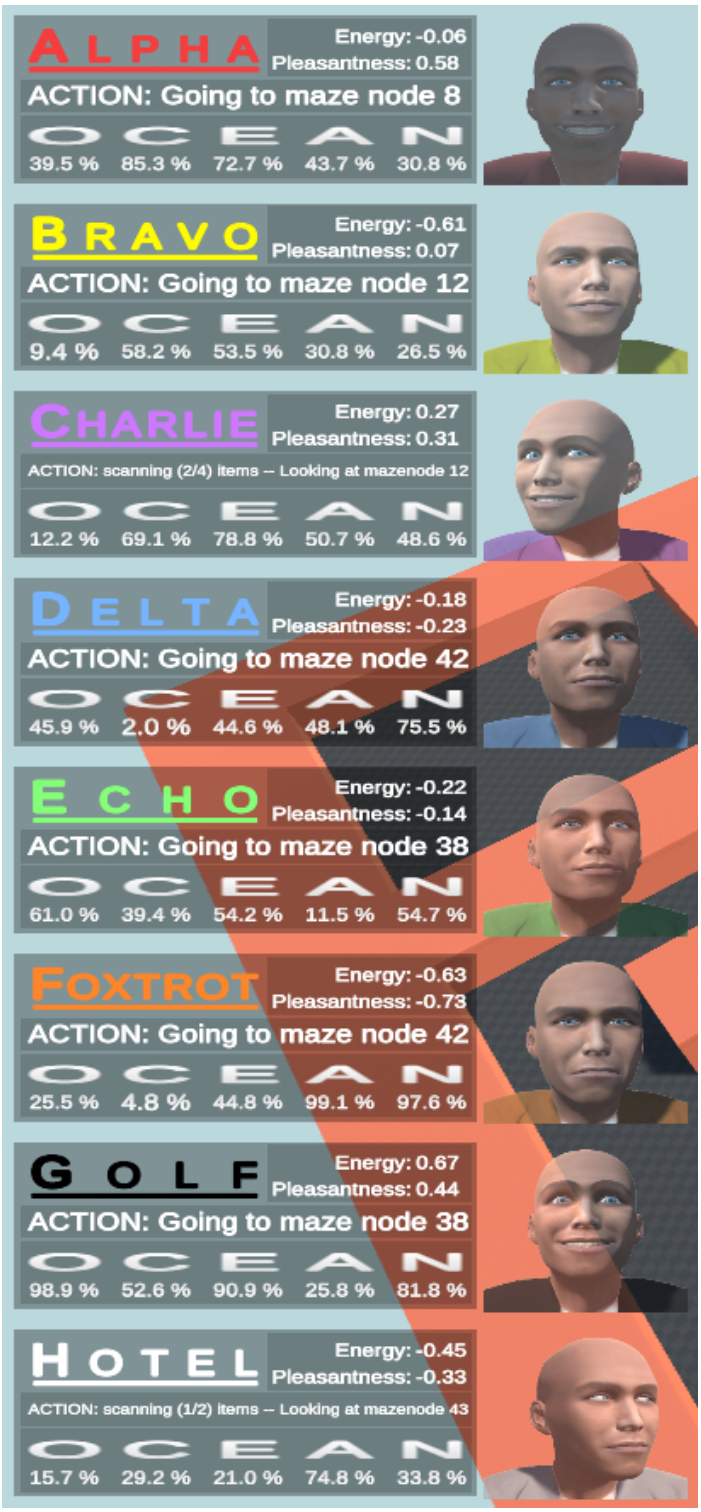

Figure 7.2: A close-up view of the NPC status user interface elements, including the view of each NPC's face. This shows the procedural facial animation for each NPC, which is determined by their mood.

Trait Versus Distance to Nearest NPC **r** Correlation	
trait	***r** Correlation*
Openness	-0.0142
Conscientiousness	-0.0275
Extraversion	-0.1309
Agreeableness	-0.0305
Neuroticism	-0.0771

Table 7.3: The correlation between each personality trait and the average distance to the nearest NPC, over time. The correlation value, r, is the Pearson product-moment correlation coefficient.

7.5 The Effectiveness of our Procedural Animation Method

While not the main purpose of this work, the procedural animation for our NPCs worked exceedingly well. Figure 7.2 offers a closer view of the facial expressions for each NPC. In the figure, it is easy to visually discern the difference in facial animation between each NPC despite our quite simple set of mood blendshape targets. As a proof of concept, this method of using a fluctuating mood to produce procedural animation is quite a success. However, this could be improved further to include more variety in animation, which is covered in chapter 8.

Chapter 8

FUTURE WORK

8.1 An Alternate Mood Model

Another popular model of mood is the PAD temperament model [29]. This model is very similar to the circumplex model of affect, except that the PAD model adds a third axis to the model: dominance. This changes the mood model from two dimensions to three, increasing the range of emotions that can be expressed by a mood model. This third dimension could provide more variation in both the behavior of NPCs in our model and in our method for procedural animation.

8.2 Personality Trait Facets

In our behavior model, personality traits are successfully used to modulate the behavior in NPCs. However, we believe that this modulation of behavior can be improved by including not only the five-factor model, but also the six facets for each of the five personality traits [20]. By including the trait facets, our behavior model would be able to simulate a more granular range of behaviors.

8.3 Procedural Animation

As a byproduct of the fluctuating mood of our behavior model, we were able to produce procedural animation by mapping blendshapes to positions on our mood graph. While procedural animation in this work was not the main focus of this work,

we believe that future research could build on our procedural animation method. We identify four areas for improvement on our procedural animation method: use of the PAD temperament model, more emotion blendshapes, a more sophisticated blending function, and body language animation.

The PAD temperament model [29] adds a third dimension to the circumplex model of affect, which would increase the range of emotion that characters could express. Emotion blendshape targets could be mapped to three-dimensional positions on the PAD model.

In our model, we only use four emotion blendshape targets. Fidelity of expression could be improved by identifying more emotional states, plotting them on the mood graph, and creating blendshape targets for them.

The blending function for our procedural animation method was quite rudimentary. The blending function first determines which quadrant the mood is in, then blends between the two blendshapes that bookend the quadrant. If more emotion blendshapes were added to the model and/or if the the PAD temperament model were to be used to replace the circumplex model of affect, a more sophisticated blending function must be designed. This new blending function must be able to identify the closest set of emotional blendshape targets and blend between them accordingly.

Our procedural animation method only animated the face of our NPCs. Future work could explore how the mood model could be used to produce procedural animation for the whole body. For instance, step width, arm swing, head tilt, and posture could be modulated using real-time information from the mood graph.

8.4 Simulation Environment

Our maze simulation environment served its purposes well by providing a simple, structured environment in which to test our behaviour model. In our maze simulation environment, we are able to observe a correlation between a change in personality traits and the actions of our NPCs. In order to take the next step towards our behavior model being realized within a published game, our behavior model should be tested in different environments. We believe the best environment to test our model in next is a role-playing game (RPG) environment, since we believe our model is best suited to be used in RPGs.

If a suitable RPG environment can be created, then similar methods for data collection can be used to measure correlations between personality traits and NPC actions. Further, the RPG environment could be used to iterate upon the behavior model. Compared to our maze environment, an RPG environment would be significantly more complex. It is likely that a change in environment would shed light on improvements that could be made to the behavior model in order to make the model extensible to many types of games.

Chapter 9

APPLICATIONS

9.1 Applying our Behavior model to The Elder Scrolls V: Skyrim

Released in 2011, The Elder Scrolls V: Skyrim [5] remains an immensely popular game in the open world role-playing game (RPG) genre. Like all modern open world RPGs, Skyrim features a vast world to explore that is full of NPCs. These NPCs live in medieval towns and cities, and fulfill all of the typical roles that Warpefelt [42] provides. In Skyrim, NPCs sell items, offer quests, serve as companions to the player, speak to the player, speak to each other, and generally go about their every-day lives. While Skyrim is a great game, it is a bit outdated, especially with regards to the behavior and animation of NPCs. NPCs all behave very similarly across the vast, varied regions and collective groups. Further, they deliver dialogue with deadpan facial expression, giving a quite lifeless feel to the characters.

If Skyrim were to implement our behaviour model and method for procedural animation into their game, the NPCs would be brought to life. The predictability of Skyrim's NPCs could be erased by inserting personality into the Skyrim's NPC behavior models. By varying personality traits by region, the NPCs of Skyrim could be made to feel as though they belong to a region and culture. The emotional believability of NPCs would be greatly improved as well by implementing the procedural animation from this work, as well as the improvements suggested in chapter 8.

9.2 Applying our Behavior model to NBA 2K

While we have mainly focused on the RPG genre in this work, we believe that sports games like the NBA 2K [6] franchise could also benefit from integrating our methods for behavioral AI and procedural animation. Each player in a NBA game simulation in NBA 2K could be assigned personality traits in order to inform the decisions they make over the course of a basketball game. These traits could be used to create more organic, realistic behavior during basketball games.

Assertive and submissive behaviors could be added to the game, and changes in mood could occur as the results of the assertive/submissive behaviors. These changes in mood could then be used to affect subsequent behavior over the course of the game. For instance, an assertive player could choose to approach the player with the ball and demand they pass it to them. If they receive the pass, then the assertive player's mood will become more pleasant. If they do not receive the pass, then the player's mood will be less pleasant. This fluctuation of mood could then be used to determine further interactions over the course of the game. For instance, an assertive player whose teammates do not like to pass the ball to could become a 'black hole', where whenever they do get a chance to hold the basketball, they don't ever pass to their teammates.

Chapter 10

CONCLUSION

Our behavior model uses personality traits and mood to determine the actions of NPCs in our maze simulation environment. A novel, greedy approach to behavior state traversal has been achieved by incorporating a mood-maximizing algorithm. Actions, or behavior states, can be used to generate a change in mood, which represents how the completion of the action may affect the mood of the NPC. A desired mood for each NPC can be generated based on personality traits, and then the mood-maximization is achieved by choosing actions that bring the mood of the NPC closest to their desired mood.

Building off our behavior model, our method for procedural animation utilizes the fluctuating mood to produce facial animations from blendshapes. A simple blending function is used to blend between facial blendshapes depending on the position of the NPC's mood on our mood graph.

A maze simulation environment has been constructed to observe the results of our behavior and procedural animation models. A memory of the maze can be represented as an undirected graph, where nodes of the graph can be marked as visited in order to record the areas of the maze an NPC has explored.

A correlation between the variation of personality traits and the decisions of NPCs in our simulation can be observed. This correlation shows that different behaviors be achieved by modulating the personality traits of NPCs in our simulation environment. Further, this correlation sheds light on the efficacy of personality and mood models from the field of Psychology for use in NPC behavior modeling.

BIBLIOGRAPHY

[1] Adobe mixamo. https://www.mixamo.com/#/.

[2] Autodesk character generator. Autodesk. https://charactergenerator.autodesk.com/.

[3] Sensor toolkit |AI |unity asset store.

[4] Unity game engine. Unity Technologies. version 2020.1.6f1.

[5] The Elder Scrolls V: Skyrim. [PC CD-ROM], 2011.

[6] NBA 2k21. [PC CD-ROM], 2020.

[7] N. Afonso and R. Prada. Agents that relate: Improving the social believability of non-player characters in role-playing games. In S. M. Stevens and S. J. Saldamarco, editors, *Entertainment Computing - ICEC 2008*, volume 5309 of *Lecture notes in computer science*, pages 34–45. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[8] J. Bates. The role of emotion in believable agents. *Communications of the ACM*, 37(7):122–125, jul 1994.

[9] S. Bura. Emotional AI for expanding worlds. https://www.gdcvault.com/%20play/1015613/Beyond-Eliza-Constructing-Socially-Engaging.

[10] A. Chowanda, M. Flintham, P. Blanchfield, and M. Valstar. Playing with social and emotional game companions. In D. Traum, W. Swartout, P. Khooshabeh, S. Kopp, S. Scherer, and A. Leuski, editors, *Intelligent*

Virtual Agents, volume 10011 of *Lecture notes in computer science*, pages 85–95. Springer International Publishing, Cham, 2016.

[11] P. Costa and R. Mccrae. Neo PI-r professional manual. *Psychological Assessment Resources*, 396, jan 1992.

[12] J. M. Digman. Personality structure: Emergence of the five-factor model. *Annual review of psychology*, 41(1):417–440, jan 1990.

[13] H. J. Eysenck and S. Rachman. *The Causes and Cures of Neurosis (Psychology Revivals): An introduction to modern behaviour therapy based on learning theory and the principles of conditioning*. Routledge, nov 2013.

[14] J. Georgeson. NPCs as people: Using databases and behaviour trees to give non-player characters personality, 2005.

[15] J. Georgeson and C. Child. NPCs as people, too: The extreme AI personality engine. *arXiv:1609.04879 [cs]*, sep 2016.

[16] L. Gruenwoldt, M. Katchabaw, and S. Danton. CREATING REACTIVE NON PLAYER CHARACTER ARTIFICIAL INTELLIGENCE IN MODERN VIDEO GAMES. page 8.

[17] H. Gómez-Gauchía and F. Peinado. Automatic customization of non-player characters using players temperament. In S. Göbel, R. Malkewitz, I. Iurgel, D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, and G. Weikum, editors, *Technologies for interactive digital storytelling and entertainment*, volume 4326 of *Lecture notes in computer science*, pages 241–252. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[18] M. Hashemian, R. Prada, P. A. Santos, and S. Mascarenhas. Enhancing social believability of virtual agents using social power dynamics. In *Proceedings of the 18th International Conference on Intelligent Virtual Agents*, pages 147–152, New York, NY, USA, nov 2018. ACM.

[19] I. Horswill. Lightweight procedural animation with believable physical interactions. *IEEE Transactions on Computational Intelligence and AI in Games*, 1(1):39–49, mar 2009.

[20] O. John and S. Srivastava. The big-five trait taxonomy: History, measurement, and theoretical perspectives. In *Handbook of personality: Theory and research*, volume 2, pages 102–138. Guilford Press, New York, 1999.

[21] E. Komulainen, K. Meskanen, J. Lipsanen, J. M. Lahti, P. Jylhä, T. Melartin, M. Wichers, E. Isometsä, and J. Ekelund. The effect of personality on daily life emotional processes. *Plos One*, 9(10):e110907, oct 2014.

[22] P. J. Lang. The emotion probe: Studies of motivation and attention. *American Psychologist*, 50(5):372–385, 1995.

[23] J. P. Lewis, K. Anjyo, T. Rhee, M. Zhang, F. Pighin, and Z. Deng. *EUROGRAPHICS 2014 / S. Lefebvre and M. Spagnuolo STAR – State of The Art Report Practice and Theory of Blendshape Facial Models.*

[24] L. Li and S. Macdonnell. *An Emotion-based Adaptive Behavioural Model for Simulated Virtual Agents.*

[25] A. B. Loyall, J. Bates, and W. Reilly. Integrating reactivity, goals, and emotion in a broad agent. 1992.

[26] B. Mac Namee. Proactive persistent agents: Using situational intelligence to create support characters in character-centric computer games, 2004.

[27] D. Mackay. *The Fantasy Role-Playing Game. A new Performing Art.* McFarland.

[28] M. Mateas and A. Stern. A behavior language for story-based believable agents. *IEEE intelligent systems*, 17(4):39–47, jul 2002.

[29] A. Mehrabian. Pleasure-arousal-dominance: A general framework for describing and measuring individual differences in temperament. *Current Psychology*, 14(4):261–292, dec 1996.

[30] A. Mehrabian. Pleasure-arousal-dominance: A general framework for describing and measuring individual differences in temperament. *Current Psychology*, 14(4):261–292, dec 1996.

[31] T. Mitchell. Believable agents: Building interactive personalities. pages 97–123, 1997.

[32] J. Nolan. Westworld, oct 2016.

[33] W. T. Plant. Longitudinal changes in intolerance and authoritarianism for subjects differing in amount of college education over four years. *Genetic Psychology Monographs*, 72(2):247–287, 1965.

[34] J. Posner, J. A. Russell, and B. S. Peterson. The circumplex model of affect: an integrative approach to affective neuroscience, cognitive development, and psychopathology. *Development and Psychopathology*, 17(3):715–734, 2005.

[35] T. Rist and M. Schmitt. AVATAR ARENA : AN ATTEMPT TO APPLY SOCIO-PHYSIOLOGICAL CONCEPTS OF COGNITIVE CONSISTENCY IN AVATAR-AVATAR NEGOTIATION SCENARIOS. 2002.

[36] D. Rosen. Animation bootcamp: An indie approach to procedural animation.

[37] J. A. Russell. A circumplex model of affect. *Journal of Personality and Social Psychology*, 39(6):1161–1178, 1980.

[38] N. Sanford. Introduction. *Journal of Social Issues*, 12(4):3–12, oct 1956.

[39] M. Sharma, S. Ontañón, M. Mehta, and A. Ram. Drama management and player modeling for interactive fiction games. *Computational Intelligence*, 26(2):183–211, may 2010.

[40] L. H. Stewart. Change in personality test scores during college. *Journal of Counseling Psychology*, 11(3):211–220, 1964.

[41] H. Verhagen, M. P. Eladhari, M. Johansson, and J. McCoy. Social believability in games. In D. Reidsma, H. Katayose, and A. Nijholt, editors, *Advances in Computer Entertainment: 10th International Conference, ACE 2013, Boekelo, The Netherlands, November 12-15, 2013. Proceedings*, volume 8253 of *Lecture notes in computer science*, pages 649–652. Springer International Publishing, Cham, 2013.

[42] H. Warpefelt. The non-player character : Exploring the believability of NPC presentation and behavior. 2016.

[43] H. Warpefelt, M. Johansson, and H. Verhagen. Analyzing the believability of game character behavior using the game agent matrix. page 11.

[44] H. Warpefelt and H. Verhagen. A model of non-player character believability. *Journal of Gaming & Virtual Worlds*, 9(1):39–53, mar 2017.

[45] D. G. Williams. Effects of psychoticism, extraversion, and neuroticism in current mood: A statistical review of six studies. 11(6):615–630.

[46] Y. Zhang and L. Li. A personality model based on NEO PI-r for emotion simulation. *IEICE transactions on information and systems*, E97.D(8):2000–2007, 2014.