GLaDOS

GLaDOS: INTEGRATING EMOTION-BASED BEHAVIOURS INTO NON-PLAYER CHARACTERS IN COMPUTER ROLE-PLAYING GAMES

BY
GENEVA SMITH, B.Eng.

A THESIS
SUBMITTED TO THE DEPARTMENT OF COMPUTING & SOFTWARE
AND THE SCHOOL OF GRADUATE STUDIES
OF MCMASTER UNIVERSITY
IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF APPLIED SCIENCE

Master of Applied Science (2017) McMaster University
(Computing & Software) Hamilton, Ontario, Canada

TITLE:	GLaDOS: Integrating Emotion-Based Behaviours into Non-Player Characters in Computer Role-Playing Games
AUTHOR:	Geneva Smith B.Eng. (Software Engineering & Game Design), McMaster University, Hamilton, Canada
SUPERVISOR:	Dr. Jacques Carette
NUMBER OF PAGES:	xiv, 182

Lay Abstract

Realistic video game characters are a desirable game component to increase a game's value. Even if the game's ending does not change, realistic character behaviours encourage players to replay a game multiple times to see what happens along the way. This is closer to tabletop games where players know the game's outcome, but still play because no two sessions are alike. Despite its advantages, few developments have been made towards realistic game characters. An easily recognizable factor of human decision-making and behaviour is emotion and integrating emotion into character design is one way to improve their realism. The GLaDOS system is a proof-of-concept product that incorporates psychological models of emotion into its design. To test its impact on player engagement, the system was implemented as an extension for the popular computer game *The Elder Scrolls V: Skyrim*. Preliminary test results are promising and show that further development could prove fruitful.

Abstract

Non-Player Character (NPC) believability is a game aspect that can be exploited to increase a game's replayability, but little research has been conducted on the topic. One method for enhancing a NPC's believability is to integrate human-like behaviours into their design, so that they react to players in a realistic and interesting way. A large part of human behaviour can be explained by their emotions; therefore it was selected as the inspiration for the GLaDOS system.

Two psychological theories of emotion, Lazarus's cognitive appraisal and Plutchik's psycho-evolutionary synthesis, guided the design of the GLaDOS system, although several components are not unique to these theories. An implementation of the design was created as a "mod" for the popular CRPG *The Elder Scrolls V: Skyrim* to test its feasibility within the context of a commercial game. This task required an additional psychological model, PAD space, to map appraisal values to emotion codes and intensities. Feasibility testing was done via a user study to determine if the GLaDOS system increases player engagement when compared to the original game. While the objective analysis found that there were no significant differences between the two versions, subjective participant responses expressed a strong affinity for the GLaDOS system. Since player engagement is inherently subjective, it is encouraging to see positive responses from participants. This indicates that the GLaDOS system, and NPC believability in general, is one aspect of video games that has the potential to increase a game's replayability and should be investigated further.

For Opa
Thanks for everything

Acknowledgements

This thesis would not be possible without a very dedicated group of people. I would like to thank Dr. Carette for forcing me to re-evaluate many of my choices throughout the development process and keeping lab meetings entertaining. Although he could not stay the whole time, I would also like to thank Dr. Teather for teaching me how to run proper user studies and showing me how interesting the human-computer interfaces field is (with the exception of bezels – no more bezels for me, thanks).

It is always good form to acknowledge your sponsors. Thanks Mom for kicking me into gear when I needed it and always making sure I am well fuelled, and thanks Dad for always being happy with my progress and forcing me to take movie breaks. Thank you Arnie for coming to pick me up when it is raining out and the rest of my family for always being interested in what I am doing – the Hamilton Lewandowskis, the Nova Scotian Lewandowskis, the Texan Lewandowskis, and the honorary Lewandowskis. A big thank you to Oma and Opa for always motivating me to do well in school and keeping an ample supply of keilbasa handy.

Finally, any well-oiled machine needs a pit crew. Thank you Omar for pretending to be clueless to make me feel like I know what I am doing. Thank you George for keeping Omar in line and making sure I am always up to speed with all the new games that keep being released when I am actually working. Thank you Warren for being a *Super Genius* and blinking lights really fast. Thank you Krystein for getting us to explore an uncharted city. Thank you Adam and Sasha for being my supervisor support network. Thanks to all the members of the McMaster IEEE Student Branch for giving me a place to hide out in. And a special thank you to Devin for all his support and patience throughout this journey. It took a while, but we made it (Side note: your impression of a tea kettle still brings joy to this day).

Contents

List of Figures

List of Tables

Definitions and Abbreviations

Definitions

Agency The capacity to act independently and make choices

Agent An autonomous computer system that observes and reacts to its environment; an NPC is a type of agent

Boss An opponent NPC with capabilities that exceed those of other opponent NPCs previously encountered by the player; defeating them yields rewards, such as items or game narrative advancement

Build In a dungeon crawler game, a build describes the collection of attributes and abilities that comprise a particular character

Cell In video game design, levels are built on a grid layout; one unit of this grid is a cell

Challenge With respect to video games, a challenge is a set of goals presented to the player that they are tasks with completing; challenges can test a variety of player skills, including accuracy, logical reasoning, and creative problem solving

Cognition The mental action or process of acquiring knowledge and understanding though thought, experience, and senses

Dialogue Tree A game mechanic that allows players to select conversation points that can result in different responses from, or affect future interactions with, and NPC

Dungeon Master A player that is responsible for arbitrating and moderating a role-playing game

Emergent Narrative
: Stories that are not authored by people, but are generated from player interactions and the underlying systems that govern gameplay

Escapism
: Seeking distraction and relief from reality by indulging in entertainment or fantasy

Experience Points
: A measure of an avatar's growth; accumulating a specific number of points resulting in an avatar's level advancement

Game Designer
: A general term referring to anyone whom participates in the creation of a game

Game Lore
: The foundation of a game's narrative, including world mythology and legend, history, and social structures

Game World
: The fictional universe associated with a game, including the environment, peoples, lore, and history

Genre
: A family of games that have common game challenges and scenarios

Level Design
: A stage in the game development cycle that generates the game environment and associated challenges

Multi-Player Game
: A game designed for two or more human players; computer controlled characters might be present

Player Experience
: An experience, designed into a game, that is presented to the player; each player experience is unique due to the players and how they interpret the design

Quest
: An in-game task assigned to the player that might be mandatory to proceed

Replayability
: Describes a game's potential for continued play after its main game challenges are completed the first time

Single-Player Game
: A game designed for a single human player; computer controlled characters might be present

Social Learning
: Learning that occurs through observation or direct instruction in a social context

Suspension of Disbelief
: A willingness to suspend one's rational thinking and logic in favour of enjoyment

Thalamus
: A brain structure responsible for sensory processing and motor signals

Turing Test
: A test for computer intelligence with the goal of tricking the human evaluator such that they are unable to distinguish the computer's responses from those of a human

Upgrade
: In games, upgrades refer to the improvement of equipment and items either by enhancement or replacement

Visceral
: Relating to the internal organs, especially those found in the trunk of the body

Video Game
: An electronic game requiring human interaction to generate visual feedback that is displayed on a video device such as a TV or computer monitor

Abbreviations

AI
: Artificial intelligence

CRPG
: Computer Role-Playing Game

DM
: Dungeon Master

GUI
: Graphical User Interface

HP
: Hit Points, normally referring to a game character's health

LOS
: Line Of Sight

MMORPG
: Massively Multiplayer Online Role-Playing Game

Mod
: Modification, specifically referring to player generated content

NPC
: Non-Player Character

PC
: Personal Computer

PLG	Procedural Level Generation
RPG	Role-Playing Game
UI	User Interface

Chapter 1

Introduction

Many video games lack the appeal or replayability of traditional board games because of limited variation between consecutive playthroughs. This can be caused by a number of factors, including linear story-telling (Riedl, 2005), the reuse of game challenges, and predictable Non-Player Character (NPC) behaviours (Jacobs *et al.*, 2005; de Jong *et al.*, 2005; Spronck, 2005). Traditional board games have a high replayability value due to interactions between human opponents, which can react and evolve over the duration of a game, an aspect absent in many video games. The absence of other human players could be partially alleviated in other aspects of a video game through the use of artificial intelligence (AI). In the game industry, AI typically refers to the systems used to create computerized opponents, but also includes methods for strategy formulation, pathfinding, natural language parsing and generation, pattern recognition, and the simulation of people and creatures (Adams, 2010). There are tools that could be used to build more sophisticated game AI, such as neural networks and cognitive systems, but many developers do not take advantage of them. This can be attributed to risk management, as well as time, knowledge, and resource constraints.

Despite low adoption rates of new AI techniques in the game industry, substantial research has been done towards game AI in academia that can learn combat strategies based on past games and stored models (Bakkes *et al.*, 2005; Marthi *et al.*, 2005; Maclin *et al.*, 2005; Molineaux *et al.*, 2005; Sanchez-Pelegrin and Diaz-Agudo, 2005; Ponsen *et al.*, 2006; Schadd *et al.*, 2007; Ponsen *et al.*, 2007; Ponsen, 2004; Spronck *et al.*, 2004b; Bakkes *et al.*, 2009) and AI that scales the difficulty of game challenges to match the player's displayed skill (Spronck *et al.*, 2004a; Bostan and Öğüt, 2009; Andrade *et al.*, 2005; Aponte *et al.*, 2009; Hunicke and Chapman, 2004; Spronck *et al.*, 2003). Comparatively little research has been conducted on other factors that can influence a human player for use in video games such as the inclusion of self-interest and attention in decision-making strategies (Gal *et al.*, 2005; Kondeti *et al.*, 2005).

This might be due to a difference in game focus. For example, Computer Role-Playing Games (CRPGs) often use AI systems to improve their combat and exploration components, while ignoring the socialization aspect found in traditional Role-Playing Games (RPGs). However, part of the appeal of RPGs is to take on a persona and experience the reactions of other players to the chosen role.

The successful integration of an AI technique into a game heavily depends on the type of game under development. Games can be grouped into families, called genres, which share common game challenges, mechanics, and scenarios. It might not be possible to design a universal emotion processing AI that works well for all game genres. Therefore, CRPGs were chosen to narrow the design focus.

The RPG genre began with the popular tabletop game *Dungeons & Dragons* where a Dungeon Master (DM), another player, lead a group of players through a fictional scenario. Unlike their tabletop versions, CRPGs are usually single-player games where the role of other human players is assigned to computerized agents. Even though CRPGs are easier to manage and are more convenient for single players, the socialization aspect is hindered due to the absence of other human players. Unlike human players, story management and NPC behaviours are limited by the game script and are unable to adapt well to the current situation or to the player's actions. Computers are now able to handle more simultaneous tasks of a higher complexity, and this will only continue to improve with further technological improvements. This presents a chance to improve the player's experience by creating more sophisticated AI.

Similar to their tabletop counterparts, CRPGs are appealing to players that enjoy taking on a role in the game world which guides their decisions within it. Despite being a common element for players to interact with, NPCs often do not have a mechanism for tailoring encounters to the player's chosen role that is comparable to humans. The decision to ignore advances in computing that could be used to create socially adaptive NPCs does not match the adoption rate of other technological advances for video games, such as graphic and audio improvements.

When designing NPC behaviours, game designers typically define a discrete number of situations that involve NPC reactions to the player, and design appropriate behaviours based on those situations. This approach is impractical for many CRPG designs due to the overwhelming number of game scenarios that a player might encounter as a result of their in-game choices. Due to the growing scope of CRPGs resulting from commercial technology developments, it is becoming more unreasonable to identify every game scenario that includes the player and NPCs. This low-agency NPC design is detrimental to CRPGs in particular because they are designed to allow the player to act out a role. If an NPC is unable to react to the player in a manner that matches

the player's chosen role, they could cause a loss of game engagement and lower the game's overall replayability value. High-agency NPC designs, where the burden of deciding how to react to the player is redistributed to computerized decision-making, is one method for managing the increasing scale of game development while also reinforcing the role-playing aspect of CRPGs by creating more believable behaviours in development times comparable to low-agency designs.

There are many ways to create high-agency designs to improve an NPC's believability, but it is unlikely that a general solution can be created due to the variety of potential influences. This makes it necessary to select one aspect of NPC agency to constrain the design space. NPC socialization behaviour is a form of computerized decision-making, so it should be possible to model and build a system that controls NPC behaviours based on their personalities, social status, and current emotion state. These behavioural aspects can collectively influence the believability of an NPC. Human emotion is a good starting point to explore more believable NPC creation due to their influence over other aspects of human behaviour and thought process. It is also a good basis for believable NPC behaviours because emotional displays are easily noticed and interpreted without additional in-game training.

While methods for improving NPC believability have the potential for increasing the replayability value of video games, very little research has been done to find reliable techniques and processing structures to improve NPC agency outside of explicit game challenges such as combat and planning. This could be partially attributed to a lack of known guidelines for specifying subjective, non-deterministic decision-making processes. It is unlikely that a single method can be created to encompass every aspect of NPC believability, so the problem scope was restricted to enabling the creation of more realistic NPCs responses to their surroundings.

This thesis presents the GLaDOS system, named after one of the main characters in Valve's *Portal* games, an AI system for games to improve an NPC's believability via the assignment of emotion simulation mechanisms to influence the NPC's behaviours. System specifications were defined by combining elements from psychological models of emotion, software engineering techniques, and game design principles. Even though it was based on psychological models, it was designed to be easily understood by non-experts so that it appears to be more reliable and easy to use. Another goal of the GLaDOS system was a modular design such that it can be easily extended to suit the needs of different game designs while also being integrable into exiting games. Performance was also a concern during the GLaDOS system's development so that it would be more acceptable to include in the computationally demanding game environment. The GLaDOS system does not currently contain any unpredictable computations, so it can be systematically tested for errors to

confirm its stability after the game's release. The development of the GLaDOS system resulted in several contributions including:

- A brief analysis of replayability and methods for increasing its value with respect to different CRPG types
- A loose taxonomy of NPC types based on their assigned game roles, the role of agency in an NPC's design, and the examples of typical NPCs reactions to players in each of the CRPG types
- An overview of current game AI design approaches used in industry and an analysis of emotion simulation systems created in academia
- An examination of psychological models using a software engineering approach to determine if they could be used in a software design specification
- The design and documentation of a computerized architecture based on psychological models; selection criteria included ease of digitization and the ability to extend architecture components
- An implementation of the proposed design demonstrating its feasibility as a game design tool for the specified host game, *The Elder Scrolls V: Skyrim*
- The design and execution of a user study to determine if the resulting system improved player engagement and enjoyment; following the study, participants expressed an interest in further play sessions with the GLaDOS system

Since this is a multi-disciplinary task, several key concepts must be explored before a specification for the GLaDOS system can be created. Game design background knowledge discussed includes the types of CRPGs and how replayability is approached in each, and NPC types and agency as it relates to reacting to the player. The current state of NPC believability implementations and approaches as it pertains to emotions is explored by examining both industry methods for AI creation, and academic approaches towards emotional systems. Analysed academic system included design for research as well as games. After the foundation has been set, the GLaDOS system design began by examining well-known psychological models of emotion to determine which, if any, could be used to create a computerized model for video games. The selected theories became the basis for the GLaDOS design itself, which is presented as a game-agnostic software architecture to improve its portability between applications. To test the proposed design for feasibility and to

ensure that it produced the desired results, a proof-of-concept system was implemented for Bethesda's *The Elder Scrolls V: Skyrim* on PC. A short user study was conducted to collect preliminary reactions from people who have previously played *The Elder Scrolls V: Skyrim* to determine its viability as a game element.

Although the GLaDOS system was designed to improve a game's replayability, there is no direct way to measure this quantity because it is heavily influenced by the player's subjective perception. Instead, indirect measurements, or indicators, of player engagement were used to determine if the GLaDOS system has the potential to increase a game's replayability value. One factor that affects replayability, player engagement is a measure of how engrossed a player is in a game which can be used to predict how long they will play in a single session. This can be used to partially describe a player's enjoyment which is one of the biggest factors affecting game replayability. The collected data proved to be promising, demonstrating that further exploration into this type of AI development could be beneficial in future game projects.

Chapter 2

Computer Role-Playing Games

Video games have the potential to provide a means of wish fulfilment or escapism (Kremers, 2009), so it is important to seriously consider the user experience in a game's design. The type of wish fulfilment desired is not universal – not everyone will find the same experiences enjoyable. This means that designers must be aware of their target audience to ensure that they include expected elements and challenges. However, if a game wants to appeal to as many different players as possible, it must include a variety of game challenges without making them mandatory. Role-playing games (RPGs) are unique because they provide the player with a loose framework, giving them more freedom regarding how they choose to play. However, a successful RPG must have a core focus, which means that it will favour at least one type of player preference over the others.

In traditional pen and paper RPGs, one player is assigned the role of dungeon master (DM). They are responsible for running and arbitrating the game in addition to the creation of the game world itself. This includes level design, non-player character (NPC) generation, and the formulation of quests and challenges. This arrangement makes it possible for players to explore actions and game locations that are not part of the original design because the DM can easily extend their world to accommodate situations that they had not considered. In computer role-playing games (CRPGs), the role of the DM is shared among several groups, including game designers (content creation) and software (running and arbitrating). This enables players to enter a predesigned game world when they want rather than waiting for it to be designed for them each time. It is also easier for players to visualize the world due to the addition of graphics and audio. However, CRPGs are most commonly designed as single-player games[1], which results in several social activities being lost in the transition from pen and paper versions, including player conversations,

[1]Massively Multi-Player Online Role-Playing Games (MMORPGs) such as Blizzard's *World of Warcraft*, are exceptions to this observation.

team-oriented behaviour, and impromptu competition. There is also a lack of accepted models that are designed to support the DM's capability to extend and alter the game to suit the players. For example, one method used to adapt a game to the player is to make most in-game challenges optional because it allows players to choose which aspects of the game they want to play. However, this design decision does not have any reliable rules or models that can be used to produce the same result in different games.

2.1 Types of Computer Role-Playing Games

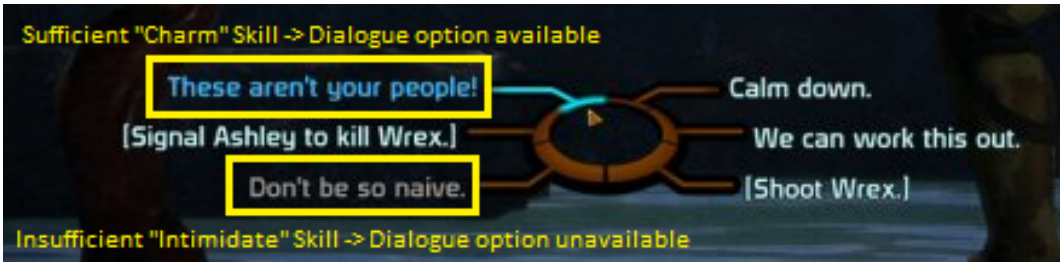
All CRPGs strive to allow players to decide how they would like to progress through the game by developing their own role, a task carried over from traditional RPGs. The player's chosen role is expressed in the game world through their avatar, which they can develop over time by distributing "experience points" collected by completing game challenges (Adams, 2010). While it was possible to develop a narrative, free-form, and mechanically strong avatar in traditional RPGs, this multi-faceted developed is currently too complex to be fully encompassed within any one CRPG satisfactorily. Therefore, game designers have made calculated decisions to ensure that the aspects that their target players enjoy are still prominent in their CRPG equivalent. Key design decisions are made based on what type of experience that designers want to create for the player through the game mechanics (Thiboust, 2013). This leads to the distinction of three basic CRPG classes: narrative, sandbox, and dungeon crawler.

2.1.1 Narrative

The focus of narrative CRPGs is game world and story, making them a good choice for players that enjoy story-driven games and meeting interesting characters. Players are encouraged to develop their avatar as a character-driven entity such that in-game choices, often in dialogues with other characters, are directly linked to their chosen character type. Key points in the story appear as explicit choices, and often require the player to make a social or moral choice that affects late-game progression. Choices that a player has made in these sections often accumulate as morality points, with players skewing towards a "good" character or an "evil" one. This type of mechanic is not new, and has appeared in games as far back as 1985[2]. However, narrative CRPGs became more prevalent in the 2000s. More recent narrative CRPGs commonly use dialogue trees to integrate player choices, making them easier to merge with the game narrative.

[2]Nihon Falcom's *Xanadu: Dragon Slayer II*

(a) Dialogue related skills



(b) Resulting dialogue options

Figure 2.1: Dialogue skills and options in *Mass Effect*

As part of the normal levelling progress that players make in RPGs, additional skills directly relating to available dialogue options might be present to improve the player's chance of success in critical game scenarios. If a morality system is implemented, the availability of such skills is linked to the avatar's moral standing. In BioWare's *Mass Effect*, the player has the option of persuading or killing a team mate in order to progress the narrative, but two options only become available if the player has sufficient skill points in "Charm" or "Intimidate" (Figure 2.1).

Another common element of a narrative CRPG is the inclusion of different game endings that are also dependent on player choices. BioWare's *Mass Effect 2* is a good example of this mechanic (Figure 2.2). The success of the final mission has many factors, including crew loyalty and ship upgrades. The narrative focus is strengthened here because the ability to influence these factors only becomes possible by speaking with crew members and undertaking additional side missions that explore their personalities and background.

2.1.2 Sandbox

Player freedom is the focal point of sandbox CRPGs, which are designed for players that like to explore and experiment. Players are usually presented with a number of initial avatar choices, such as visual appearance and initial statistics, before being given the main quest. However, players do not have to complete this quest to progress the game. They can opt to ignore it entirely in favour of different side quests, self-directed exploration, or their own goals. Allowing the player to go through the game world however they want enables

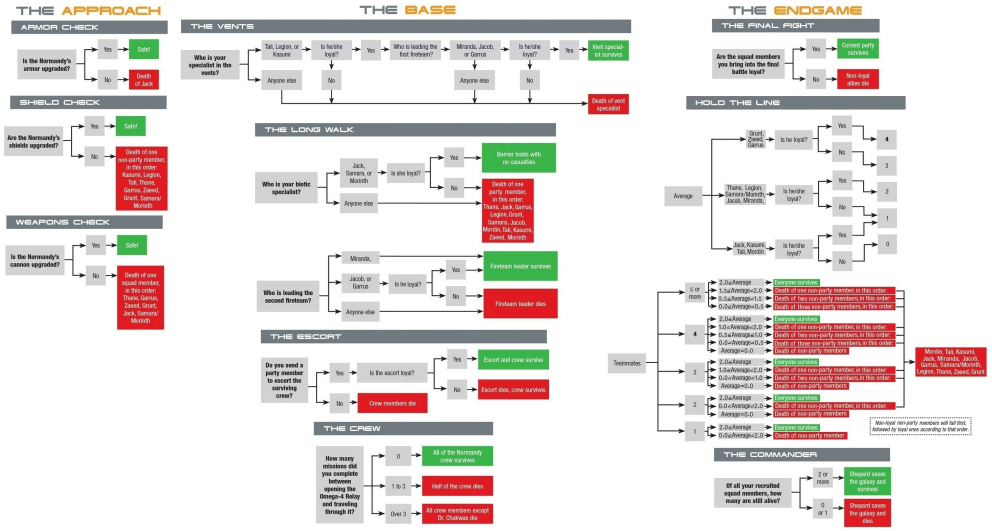


Figure 2.2: BioWare's *Mass Effect 2* Player Choice Consequences

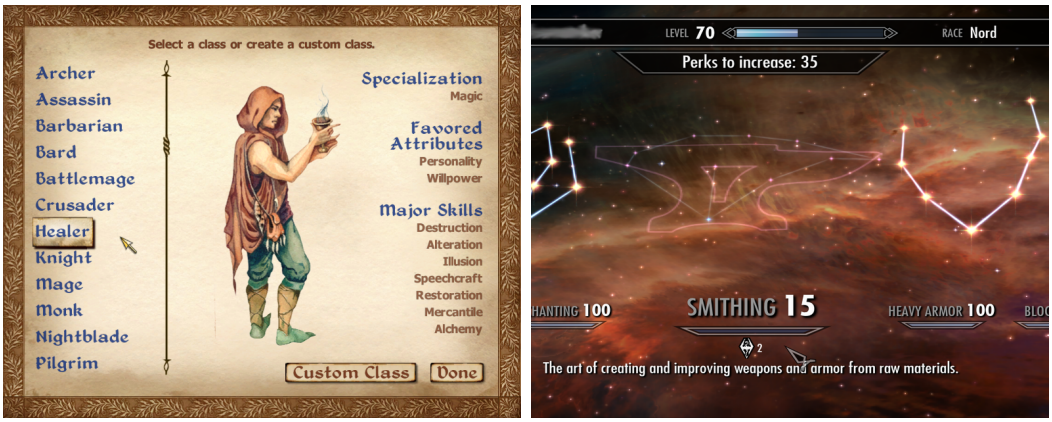
them to make their own story and try out different in-game challenges without restriction. To support this, many sandbox CRPGs have a more robust navigation and movement system.

This freedom also allows players to develop their avatar in potentially suboptimal ways, such as learning abilities that their avatar does not benefit from. This type of avatar development is especially apparent in games with avatar classes which assign bonuses to particular skills such as *The Elder Scrolls IV: Oblivion*, where players can only gain levels by improving their major skills (Figure 2.3a). In *The Elder Scrolls V: Skyrim*, the traditional avatar classes were completely removed in favour of eighteen different skills divided into three types[3], allowing players to gain levels by improving any skill (Figure 2.3b). Theoretically, this new mechanic makes all skills useful to avatar development. This enhances player freedom by removing penalties previously incurred for trying skill combinations that are not traditionally used by their avatar's assigned class.

2.1.3 Dungeon Crawler

The dungeon crawler CRPG is typically favoured by competitive-minded players because it focuses on the avatar's strength and abilities. Players overcome challenges by improving their avatar's statistics, such as strength, ability set, and equipment, usually via combat where opponents award both experience points and items. A dungeon crawler might not have a story, but it should include world lore to provide a purpose or guide for avatar development, hints

[3]Warrior, Mage, and Thief



(a) Classes in *The Elder Scrolls IV: Oblivion*

(b) Skill tree in *The Elder Scrolls V: Skyrim*

Figure 2.3: Major skills and level progression in *The Elder Scroll* series

for defeating bosses, and clues for locating special items. Players also often expected to be able to customize their equipment via upgrades.

In Grinding Gear's *Path of Exile*, the player is simply told that they have been sent into exile on an island and must now survive on their own. Avatars are improved by defeating opponents, which drop items that can be sold or equipped, and award experience points. Heavy customization is encouraged due to the scale of the skill tree (Figure 2.4a), and the ability to modify equipment with gems that grant additional benefits (Figure 2.4b).

(a) Skill Grid

(b) Equipment Grid

Figure 2.4: Avatar progression in Grinding Gear's *Path of Exile*

2.1.4 Why is Action not a CRPG Type?

Recently, many CRPG titles have been labelled "action-RPGs", including the games discussed here. While the description is not inaccurate, simply calling them action-RPGs is not as descriptive as the categories named here, and can lead to confusion about the game's focus. Even from this brief analysis, it can be shown that these games differ greatly from each other.

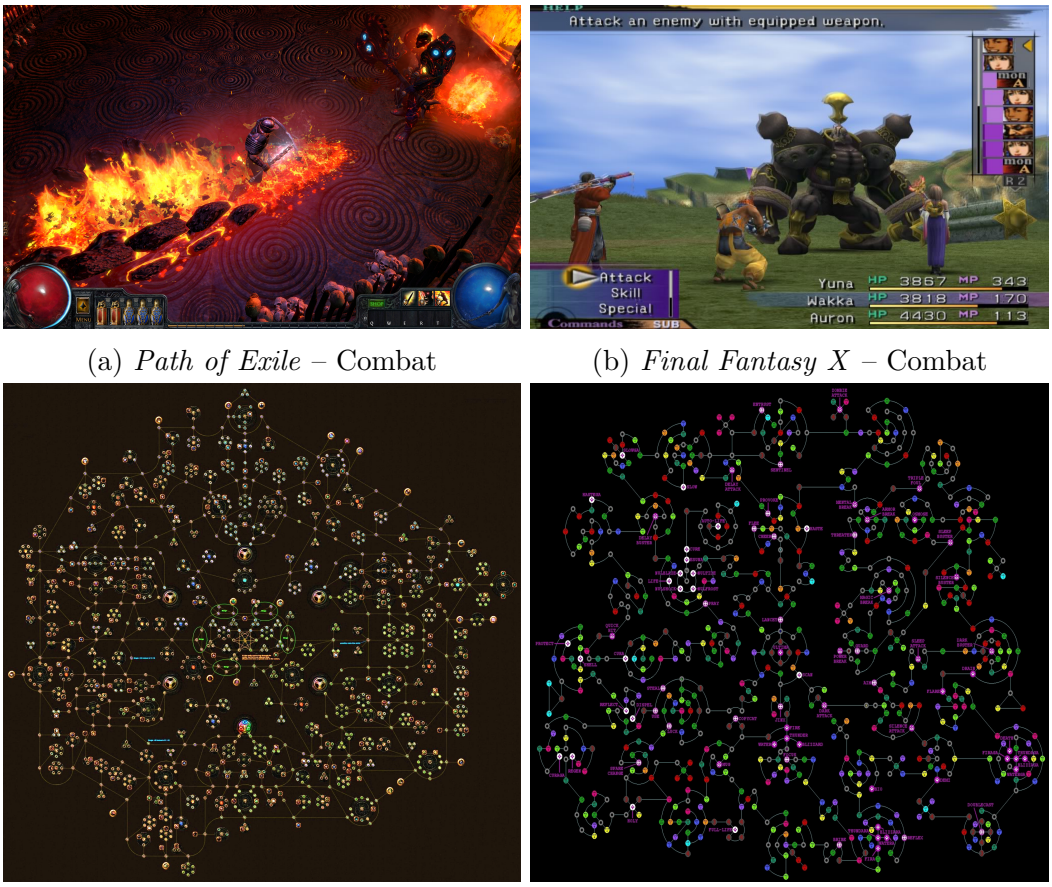
The classification of action-RPG describes how in-game combat is handled. In an action-RPG, combat is handled in real-time and players must rely on instinctive decisions and quick movements in order to succeed. In contrast, turn-based RPGs use turn-based combat mechanics, allowing players to analyse opponents and plan their tactics. It is slower, but does not rely on the player's reflexes to be successful. However, the two mechanics of "action" and "turn-based" can be used in any genre and generally divides all types of games into two groups.

Consider Grinding Gears' action-based game *Path of Exile* and Square Enix's turn-based game *Final Fantasy X* (Figure 2.5). While they do not share combat mechanics, their core gameplay focus is character progression via skill acquisition. Instead of being two different games, as their action and turn-based labels would imply, their core mechanics put them into the same category – dungeon crawler. Similarly, although the action-RPG label puts both BioWare's *Mass Effect* (narrative) and Bethesda's *The Elder Scrolls* series (sandbox) in the same category, playing each series for a short time shows that their focus is very different from the other.

2.2 Replayability

A common concern held by game designers is how to keep the player interested once they complete the main game challenges, causing them to extend their time investment in the game. This continuation of play, or replayability, is a game design concept that the GLaDOS system aims to improve. Unfortunately, there is no direct way to impact a game's replayability value because it is heavily influenced by player perception. Instead, identifying areas in each CRPG type that a player might find interesting can be exploited as potential elements to improve their engagement with the game. If a player becomes sufficiently engaged with a game, they are more likely to continue to play it after the main challenges are completed. This continuation of play is encouraged by CRPGs in particular because they encourage players to define and act out their own role in the game world. This aspect of CRPGs is only limited by the scope of each individual game's mechanics.

Most CRPGs inherently have several hours of play time before the player



(a) *Path of Exile* – Combat (b) *Final Fantasy X* – Combat

(c) *Path of Exile* – Skill Grid (d) *Final Fantasy X* – Skill Grid

Figure 2.5: Comparing the Action Dungeon Crawler (*Path of Exile*, Grinding Gear Games) and Turn-based Dungeon Crawler (*Final Fantasy X*, Square Enix) CRPGs

considers the game beaten, but might still be considered stagnant if the repeatable elements have little variation. Game designers have tried several methods to increase the replayability of CRPGs, but replayability value can also be player-directed. One method of improving replayability is through the game design itself. A common method that game designers use to include additional content and extend the shelf life of their games is a side quest. Depending on the desired experience, side quests can include non-essential narrative pieces, item collection, and more difficult combat challenges. However, side quests are often designed to be defeated once per play through, or, when they can be played many times, offer little variation between consecutive undertakings. For example, radiant quests, introduced in *The Elder Scrolls V: Skyrim*, are procedurally generated for each of the factions that the player is able to join. However, only the target and location of each quest are changed. Since the

underlying mechanic remains unchanged, the replayability value is reduced because a player can grow bored of it.

Narrative CRPGs inherently have some replayability value due to player choice points, and are often replayed to see what type of impact different choices have on the game's final outcome. NPC opinions, available challenges, and even the game's conclusion are candidates for this type of gameplay extension. However, these options are typically pre-designed to ensure that any player selection still makes sense within the game's narrative. This means that there is still a point at which the player completes the game. Recently, there has been proposals for emergent narrative (Martens *et al.*, 2014; Chauvin *et al.*, 2015; Lucat and Haahr, 2015), but there currently is no indication that this work has been employed in commercial games.

Another mechanic that can improve the replayability of a game is procedural level generation (PLG), where a game level is randomly generated via a generation algorithm each time the player encounters it. One notable CRPG series that uses PLG is Atlus's *Shin Megami Tensei: Persona*. Integrating PLG into this series is successful because the randomized generation of each level is explained within the scope of the game narrative. This tactic might not be possible for many designs because of the need to integrate it into the game lore.

Player-directed replayability can be difficult to accommodate, but is potentially the best way to increase a game's shelf life by allowing players to create their own experience. Sandbox games explicitly allow players to create their own experience by allowing them to define their own objectives and goals. In *The Elder Scrolls* series, players often create their own stories and personalities for their avatars to clearly define how they will dress and act. Another means for players to personalize their experience is to define jobs outside of the scripted game, such as being a hunter (killing animals and selling their pelts) or an alchemist (growing alchemical plants, brewing and selling potions). The series does not explicitly define these roles, but provides players with the means to create the role on their own. NPC behaviour becomes an important feature in this play style, as the player normally expects some type of acknowledgement or reward for playing their role. Another facet of player-directed replayability is trying different actions to see how the game reacts. This can be done implicitly, such as trying different roles to determine which one they favour, or explicitly to find unusual behaviour or bugs. This type of player-directed replayability is especially prevalent in dungeon crawler CRPGs. A good combat customization system can potentially provide countably infinite combinations of equipment and skills, or builds, which players can create. If the opponents in the game are interesting enough, then players enjoy defeating them multiple times with different builds to see how they compare. Players might also challenge themselves by deliberately creating a substandard

build or try experimenting to find characters that are unexpectedly strong.

Some games allow players to create their own content via modifications, or mods. Mods are typically created by more technical-minded players who enjoy game creation and design as a hobby, and can include maps, textures, and scripts. Mods can extend the shelf life of a game indefinitely as long as player communities continue to create new content. However, this practice is mainly applied to PC games due to the additional publication rules of console games.

Replayability is a trait favoured by players, and therefore game designers, which can extend the shelf life of a game. It can be improved through the game's design by including side quests, narrative branches, or PLG. It can also occur after a game's release via player-generated content. Extending the life of a game can improve a player's favour towards a developer, series, or an entire genre of game. Although it does not appear to make good business sense to encourage a player to continuously play a single game, delivering high-quality content that can be enjoyed after the initial play though can improve the chances of that player consuming additional titles from the same developer. This, in turn, ensures that a game designer can continue their work. Thus far, most aspects included in CRPGs to improve their replayability are limited to those explicitly developed by the game designer. However, this approach is limited because it relies on a designer's ability to prepare a variety of situations during a game's development and does not typically allow for more scenarios to be added post-release. A common element found in many of the approaches that game designers have used previously are NPCs. They are often used to facilitate quest acquisition, provide conversation, and populate the game world where a player acts out their role. Since they are already used for a variety of tasks and are utilized in all CRPG types, NPCs are a good aspect to study to indirectly improve a game's replayability value. Therefore, the GLaDOS system focuses on improving a game's replayability via NPC interactions.

2.3 Conclusion

It is possible for CRPGs to cater to a variety of player fantasies, but the shelf life of these games is directly tied to their ability to maintain the illusion. The framework that CRPGs offer allows players to define their own experiences. However, this framework must also include incentives for players to continue to live in their chosen roles after they complete the game's main challenges. In traditional tabletop RPGs, the DM could dynamically tailor new experiences to suit the current players. However, the ability to do this easily is lost when translating the same game to a computerized format, largely due to a lack of reliable formalized methods and models.

To help streamline the player experience and narrow a game's focus, designers often choose a single feature of traditional RPGs: narrative CRPGs focus on plot and character development; sandbox CRPGs focus on exploration and experimentation; and dungeon crawler CRPGs focus on combat and avatar development. It can be argued that the narrative subtype has the shortest shelf life due to its pre-designed and scripted responses to player choices. However, steps have been taken academically towards emergent narrative experiences which could potentially result in endless replayability.

Common methods for increasing a CRPG's replayability value, such as side quests, no longer appear to be sufficient for extending gaming experiences, although they are still heavily featured in modern CRPG designs. Other methods, such as PLG, can potentially be used in any game to improve replayability, but their inclusion is dependent on the game's context and might not make sense for some game designs. Well-designed sandbox and dungeon crawler CRPGs inherently have replayability value because it is player-directed, a trait carried over from traditional RPGs. Player-directed replayability is often based in role experimentation, so it is important that a game reacts dynamically to retain player interest. Player-generated content is another good way to increase the life span of a game because it allows players to create and share their own modifications for a published game. However, this method is limited in console games due to stricter publication rules. Overall, there are many different ways of improving the replayability value of a CRPG. The method chosen depends on the type of game and the target audience.

NPCs are potentially one of the best ways to provide the feedback that players require to reinforce their playing habits, especially since NPCs are often key components in CRPGs. Due to their universality as a CRPG element and their ability to interact directly with players, NPCs were selected as the implementation target of the GLaDOS system. Before further decisions can be made on the GLaDOS design, it is important to understand the different types of NPCs found in CRPGs and common tasks and roles assigned to each so that aspects of their design can be identified for augmentation.

Chapter 3

Non-Player Characters

A Non-Player Character (NPC) is any game character that the player does not have complete behavioural control over (Carreker, 2012). NPCs are traditionally used to initialize quests, advance the game's plot, and facilitate a market system for players to exchange their unwanted resources for more desirable ones. This list is not exhaustive – NPCs might also be used for additional game functions depending on the game's design. In tabletop RPGs, NPCs are created and controlled by the DM, which limits the number and variety of in-game characters used. Due to the nature of tabletop games, the DM often ends up creating basic personalities for each NPC to keep other the players engaged.

With the power of computers, video game NPCs are also used to create an ambient game world population in addition to their traditional roles. However, due to the size and scope of more extensive game worlds, many non-essential narrative NPCs, including townsfolk, guards, and common opponents, are copies of a core set models. This practice can be observed in games with at least partially open worlds, such as Ubisoft's *Assassin's Creed* series and THQ's *Saint's Row* series. This approach often results in NPCs that perform repetitive behaviours and have near-identical physical appearances to other NPCs which the player might perceive as unnatural and ultimately undermines their experience.

Some developers try to alleviate this problem by giving each NPC some unique traits but they are usually cosmetic, such as the NPC's name and colour scheme. Bethesda's *The Elder Scrolls* series is a good example of this technique. It is rare to find a densely populated game world where each NPC behaves in a unique manner. It is more common to find different classes of NPCs where the behaviour of the class differs from other classes. There are exceptions to this observation, such as Spike Chunsoft's *Danganronpa* series and Procedural Arts' *Façade*, where each NPC is completely unique. This is facilitated by having a small number of NPCs, where each one is non-trivially

required in the game narrative.

Before a GLaDOS system design can be proposed to extend NPC capabilities, the purpose for including each type of NPC must be understood. Video game NPCs can be divided into three broad categories: companion, opponent, and ambient[1]. Although it is not possible for an NPC to be assigned multiple categorizations simultaneously, it is possible for an NPC to be categorized differently over the course of a game's narrative. These changes are usually dictated by the game's narrative. A famous example is Sephiroth from Square's *Final Fantasy VII* who joins the player's team as a companion for a time before becoming an opponent.

3.1 Companion

Companion NPCs are characters that the player can form an alliance with and might be able to exert some control over. Companion NPCs are usually part of the game's story, often added to assist the player with game challenges. One common way for companion NPCs to be used is for combat challenges. For example, in *The Elder Scrolls V: Skyrim*, the player has several opportunities to recruit a House Carl (Figure 3.1a), who assists with any combat challenges that the player encounters. House Carls can also be used for non-combat purposes, such as allowing the player to carry more items in their inventory, and provide some additional dialogue to help explain their role in the game world as it relates to the player. Elizabeth from Irrational Games' *Bioshock Infinite* (Figure 3.1b) is an example of a non-combat companion. The narrative focus of the game is to escort her out of a city, which requires the player to keep Elizabeth as a companion indefinitely after meeting her. In addition to her importance to the game's narrative, she also assists the player with game challenges by providing extra resources and information.

3.2 Opponent

Opponent NPCs are in direct conflict with the player and are part of some in-game challenges. Like companion NPCs, opponent NPCs are at least partially defined within the game narrative. This can be as an individual or as part of a group affiliation that is integrated into the game world. When discussing video game opponents, many people are referring to NPCs that the player must defeat in combat challenges. Combat challenges can either be direct, such that the player takes them head on, or indirect, such as in stealth

[1]The terms "companion" and "opponent" are commonly used within the game community to describe NPC types. "Ambient" is used to describe NPCs that do not meet these two commonly used definitions.

(a) Lydia, a House Carl (Bethesda's *The Elder Scrolls V: Skyrim*)

(b) Elizabeth (Irrational Games' *Bioshock Infinite*)

Figure 3.1: Examples of companion NPCs

games like Ubisoft's *Tom Clancy's Splinter Cell: Chaos Theory* (Figure 3.2a) where players are challenged to defeat enemies without fighting them directly. However, this definition is too narrow and does not account for other types of conflict that the player might encounter. In Rockstar's *L.A. Noire*, a key challenge that the player faces is interrogation of persons-of-interest (Figure 3.2b). These NPCs can be considered opponents because they need to be "defeated" in order to access the information that they hold – the player's reward. If the player does not get enough information in a single case, which includes several persons-of-interest, they might fail the overall challenge set and receive a reduced final score or suboptimal story ending. This is the same scenario that a player encounters with combat opponents – they must defeat enough of them to proceed to the next set of challenges or to get a high score.

(a) Soldier (Ubisoft's *Tom Clancy's Splinter Cell: Chaos Theory*)

(b) Person of Interest (Rockstar Games' *L.A. Noire*)

Figure 3.2: Examples of opponent NPCs

3.3 Ambient

An ambient NPC is a non-essential character that is neither an aid nor challenge to the player, and used for a variety of purposes. Common tasks assigned to ambient NPCs are quest distribution and the facilitation of a bartering system for the player to use. Some games even integrate ambient NPCs into a game mechanic. In Ubisoft's *Assassin's Creed* franchise, ambient NPCs can be used to disguise the player to help them sneak into heavily guarded areas (Figure 3.3a). Not all ambient NPCs have a specific task however – they might be present simply to make the game world seem more realistic. Depending on the design, these types of NPCs might not be interactive. For example, the player can interact with ambient NPCs in THQ's *Saint's Row the Third* (Figure 3.3b) in a number of different ways, while ambient NPCs in Polyphony Digital's *Gran Turisimo 5* cannot be interacted with (Figure 3.3c). While a game can be designed with only companion and opponent NPCs, it is usually the ambient NPCs that breathe life into the game world and make it appear more alive.

(a) Hiding in a crowd (Ubisoft's *Assassin's Creed: Brotherhood*)

(b) Steelport Citizens (THQ's *Saint's Row the Third*)

(c) Spectators (Polyphony Digital's *Gran Turisimo 5*)

Figure 3.3: Examples of ambient NPCs

3.4 The Role of Agency in Non-Player Character Behaviours

An NPC's ability to make self-directed decisions is known as their agency, and directly impacts a game's replayability value. As part of their specification, game designers assign a set of behaviours, and behaviour-selection mechanisms to NPCs which are used to alter their behaviours in response to changes in the game environment.

When an NPC has low agency, game designers author instructions that explicitly select which behaviour NPCs use in response to specific scenarios or scenario classifications. This is comparable to a state machine, where NPC behaviours are the machine's states and changes in game world scenarios are the state transitions. Even though this approach ensures that NPCs behave as intended in each scenario, the specifications required to author pairings of scenarios and behaviours quickly becomes unmanageable as game complexity and the number of required NPCs increases. This often leads game designers to begin grouping scenarios together as a single type such that multiple scenarios lead to the same NPC behaviours. While the grouping of game scenarios often makes logical sense from a design standpoint, the interpretation of individual scenarios is usually perceived differently by players. It is common for players to notice repetition in games because they become more attuned to similarities as time progresses. This can lead to odd-looking behaviours from the player's perspective.

In comparison, an NPC with high agency is assigned a selection mechanism that collects information from the game environment and uses it to choose which available behaviour set should be used in response. Designing selection mechanisms causes game designers to think differently about their game's scenarios by forcing them to identify abstract patterns rather than concrete events. Game designers can then associate NPC behaviours with scenario patterns as opposed to individual scenarios. By enabling NPCs to pattern-match an encountered game scenario to general ones, the task of associating events to behaviours is transferred from the game designer to the NPCs. While this does not immediately appear to be different from low-agency designs, the ability to pattern-match encountered scenarios with a set of general ones alleviates the pressure on game designers to identify all possible game scenarios and manually assign resulting behaviours to each one. It is likely that a game designer can identify all potential scenario patterns that are relevant to an NPC's agency because a game has a constrained design space with strict rules. Despite a game having an unmanageable number of individual game events, they can each be encapsulated by at least one of a smaller and more manageable number of abstract event patterns. This approach also has the benefit of preventing human error during behaviour assignments and can

be expanded to include more scenarios as needed, rather than as a method of error prevention.

Increasing the agency of an NPC requires the game designer to create well-specified selection mechanisms while also forfeiting control over exact behaviour selection. This release of control might make some game designers nervous because they want NPCs to behave in a particular way in specific contexts. However, as games continue to increase in complexity it will become more difficult for designers to predict every scenario that a player will encounter. This reality makes it more practical to create NPCs with higher agency to alleviate the pressure on game designers in exchange for complete control over NPC behaviours. The design of high-agency NPCs is an effort-intensive task, which might not fit within a game designer's time constraints. The design of the GLaDOS system should help expedite this task by providing a framework for high-agency NPCs. This will reduce a game designer's workload such that they only need to provide game-specific information to the GLaDOS system to make it functional. Despite being a feasible task, the scope of a modern game is still too large for most game designers to consider the identification of all potential scenario patterns a worthwhile endeavour. Since their main purpose in a game is to interact with the player, scenarios that included player interactions were considered exclusively in the GLaDOS system design space.

3.5 Reacting to the Player

The purpose of NPCs is to engage the player either through direct interactions or by enhancing the believability of the game world. At the very least, NPCs should not degrade the relationship that the player has with the game. Despite this, NPC behaviour is one cause of broken player engagement due to their, typically low, agency. A general example of this problem is repetitive behaviour that, when coupled with near-identical visual appearances, creates an unnatural feeling. The issue becomes even more apparent when the player expects the NPCs to react differently based on what is happening around them. The degree of disturbance depends on the game, and some games might be able to consider this a cosmetic requirement. For example, the focus of dungeon crawler CRPGs is combat challenges, therefore it is not essential that any NPC react believably to the player. On the contrary – players usually expect opponent NPCs of the same type to behave in the same manner in this type of game, but it would not hurt for key NPCs, such as bosses, to behave differently outside of combat depending on the player's performance so far. Behaviours could include something as simple as different dialogue sets. This could be used to coax the player to continue to the next set of challenges if they are doing well, or to try again if they did not defeat the NPC. Even

Figure 3.4: No reaction to "famous" player in *Mass Effect*

though the application of this type of NPC enhancement is clearer for CRPGs, other types of games could benefit from augmented NPC reactions and. In some cases, this type of enhancement can become a mechanic that augments gameplay.

In narrative CRPGs, where the focus is on character development and story, NPCs should react to the player based on their role in the game narrative. This specification is usually upheld by essential story NPCs, but unlikely to be considered for non-essential ones such as ambient NPCs. For example, if the player character is an average person, players do not expect NPCs to react to the player's presence any more than they would for another non-essential NPC. However, if the player is a well-known figure, such as Commander Shepard in the *Mass Effect* series, players could expect NPCs to react differently. Unfortunately, they do not (Figure 3.4). While the additional behaviour can be omitted from the game with no critical adverse effects, their inclusion could enhance the player's connection to the game by reinforcing their importance in the game world.

The addition of reactionary NPC behaviour is non-essential to dungeon crawler and narrative CRPGs, but it would enhance the player's experience cosmetically. In contrast, the focus of sandbox CRPGs is player freedom, which makes it more critical to include at least some reactionary NPC behaviour. NPCs should be happy to see the player or afraid of them depending on how the player has developed their character, but sandbox CRPGs often have the most trouble reacting to the player. Implementing this type of behaviour using traditional development practices can quickly become unmanageable due to the range of behaviours required. The result is often the addition of a few basic behaviours that can be comedic if viewed or triggered in the wrong context. For example, in *The Elder Scrolls V: Skyrim*, if a nearby NPC is

(a) Child standing on dead guard

(b) Ignoring the player who is in a combat-ready state

Figure 3.5: Indifferent NPC Responses to the Player in *The Elder Scrolls V: Skyrim*

killed by being shot with an arrow, at least a few NPCs should be afraid or suspicious of the player if they are carrying a bow (Figure 3.5a). Similarly, most NPCs should be afraid of the player if they are wearing armour that makes them look like a demon while their weapon is drawn (Figure 3.5b).

An NPC's reactions are one way to acknowledge the player's role within the game world in addition to giving the player feedback on how they are affecting the game. The effect can be cosmetic, such as in a dungeon crawler CRPG, or it can be used to reinforce the player's role, such as in narrative CRPGs. It could even evolve into a mechanic of its own in sandbox CRPGs. No matter the size of the impact, if an NPC reacts to something related to the player in a somewhat believable manner, the player experience can be

enhanced and players might become more engaged in the game. This makes it essential for the GLaDOS system design to be based on a concept that can be universally applied to any NPC in any game. There are very few concepts that are not limited by cultural boundaries, however human emotion is known to be recognized regardless of which culture it is presented to (Ekman, 1992). This universality makes it an ideal basis for the GLaDOS system because, since it is universal across human culture, it can easily be transferred between different game designs and still be understood. The digitization of human emotion processing is not a new endeavour, and it is likely that a number of useful ideas can be gathered from past designs.

3.6 Conclusion

Essential to most video games, NPCs enable the distribution of quests, progress narrative elements, and generally make the game world feel alive. The player is aided by companion NPCs and challenged by opponent NPCs, while ambient NPCs flesh out the game world. Therefore, it is important that NPCs are designed in an efficient and realistic way such that they do not put unnecessary strain on the system while convincing the player that they are real. This requires game designers to assign a higher agency to NPCs so that they are able to adapt to the player without explicit guidance. Due to time or resource constraints, this requirement is often left incomplete or an attempt is made that does not adequately meet the specification. Even though creating high-agency NPCs appears to be a difficult task, the challenge is in forcing game designers to think differently about their game and create abstractions for pattern matching rather than specifying exact rules for NPCs to follow.

The importance of higher NPC agency depends on the game design. For dungeon crawler and narrative CRPGs, this requirement is usually cosmetic, but can enhance player engagement by reinforcing their importance to the game world. In sandbox CRPGs, this requirement becomes more critical because this game type provides more mechanisms for the player to create their own character. This goal is usually accompanied with the expectation that the game world will react at least a little differently depending on what the player chooses to do, including the player's physical appearance and moral alignment. Here, reactionary NPCs become more like a game mechanic which can improve player engagement if it is implemented well. Overall, the addition of reactionary NPC behaviour can enhance any type of CRPG by reinforcing the player's importance to the advancement of the game. Due to the variety of potential game designs, the GLaDOS system should be based on a universally recognized concept, such as human emotion. This type of design task is not new, and observing the design of similar systems will help guide the subsequent design decisions required to create the GLaDOS system.

Chapter 4

Artificial Intelligence for Non-Player Characters

There is a noticeable disconnect between the video game industry and academia when it comes to integrating more complex artificial intelligence (AI) systems into commercial applications (Yannakakis and Togelius, 2014). Since creating a good player experience is key to a successful game, many developers want to mitigate as many technical risks as possible. Despite frequent adoptions of graphics and audio quality advancements, AI advances are usually ignored due to their additional design requirements, development time, and lack of guarantee regarding expected behaviour (Graft, 2015). This results in the use of AI techniques that players can easily exploit and tire of due to their subsequently predictable behaviour – a trait that does not match with the industry's rapid development and increasing project scale. Even within the industry, it has been suggested that one of the next major commercial breakthroughs are NPCs that behave realistically, such as more unscripted responses to player actions and adapting to the changing game environment (Ponsen, 2004; Graft, 2015).

Systems for improving the believability of computer agents have been created previously for both academic (Section 4.2) and commercial purposes (Section 4.3), but they have not been widely accepted outside of the academic community. To improve the acceptability of the GLaDOS system, it is useful to review what approaches are typically used during a game's development to create NPCs before looking to academia for further design ideas and guidelines.

4.1 Artificial Intelligence for Game Design

There are three main approaches to designing AI for games: ad hoc, heuristics, and algorithms (Millington and Funge, 2009). Neither ad hoc nor heuristics are considered true AI because they utilize programming "tricks" instead of a

defined AI technique or algorithm. A good game tends to use a combination of all three approaches to achieve the desired overall effect and to manage computational speed and memory requirements effectively.

The ad hoc, or "hacking", approach is comparable to behaviourism (Watson, 1919), where studying behaviour and its construction can lead to an understanding of the object that displays that behaviour. As a science, behaviourism is no longer considered a valid approach but it still has uses in game programming to make characters "look right" with little effort or computational resources. Triggering an animation to convey emotion and choosing behaviours at random are examples of simple programs that have the potential to produce convincing behaviour if applied judicially. For example, the ghosts' pursuit logic in Namco's *Pacman* appears random, but are actually scripted so that each ghost follows strict rules. There are drawbacks to these methods, including the potential requirement for additional experimentation and testing to ensure that the expected behaviours are being produced, and these programs are often impossible to transfer between games.

Heuristics are probably the most common method used by game designers. It incorporates approximate solutions that work in most situations, similar to the way humans use general principles and folk wisdom to make snap decisions. This approach is used when complex calculations can be simplified, often down to a single number, to reduce computational speed and memory requirements. In exchange, the accuracy and robustness of the resulting behaviour is sacrificed. Common heuristics used for game AI include doing the most difficult task first, such as attacking the opponent with the highest total health, and moving in the current direction of a moving target, which can give the illusion of intelligent chasing or flocking mechanics.

Behaviours that do not change between projects are turned into algorithms, allowing programmers to recreate AI that are not easily transferable as ad hoc or heuristic solutions. This aspect of game programming is typically the target of academics because algorithms can be analysed and tested with reliably similar results. The A* path finding algorithm, minimax algorithm for turn-based games, and decision trees are examples of academically-tested algorithms that are used in video games. Collections of related algorithms can be collected into a game engine that targets a specific game aspect such as physics and animations.

These three design techniques – ad hoc, heuristics, and algorithms – are used in the design stage of a project. Implementation techniques also tend to follow simple patterns to reduce overall development time and resource use. Rule-based approaches, such as conditional scripts and state machines, are typically used to implement game AI due to their ease of use, understandability, testing, and extensibility. However, these approaches are often deterministic and force designers to anticipate every possible game state (Ponsen,

2004; de Jong *et al.*, 2005; Yannakakis and Hallam, 2005). This is an almost impossible task, especially when there are many player selection points. Additionally, due to their static nature, scripts and state machines are prone to weaknesses that can be exploited by the player or cause unreasonable results for the current in-game scenario (Spronck *et al.*, 2003; Ponsen, 2004; Andrade *et al.*, 2005). State machines in particular do not scale well, forcing designers to create simpler behaviours that become increasingly predictable (Wray *et al.*, 2005). The organization of states, transitions, and scripts is an aspect of game design that the GLaDOS system design can borrow. Although it will not be as limited as a state machine, the GLaDOS system can be designed such that game designers can define game scenario patterns as if they were states and then assign behaviour scripts or other state machines to them. Designers can then define a single calculation to determine when a behaviour transition should occur. This will reduce the scalability and prediction requirements typical in state machine designs. Using calculations with a continuous range instead of state transitions with a discrete range will also make it more difficult for players to predict and exploit the resulting outputs of the GLaDOS system.

Not all game industry approaches to AI are considered as such in academia, but a true AI design might not be necessary depending on the designer's goals. Ad hoc and heuristic methods have been successful in the past, but are becoming more unreliable as game complexity increases. Even implementation methods are becoming unreliable as the scale of commercial games continues to grow. Industry methods do still include algorithms, which are constantly studied in academia implying that there is potential for academic advances to aid in game development to help match rising complexity demands from designers and players alike. Although it does not produce the same outputs as a state machine, the GLaDOS system might appear structurally similar to game designers, making it more likely that the system will be adopted. This is because similar groups of information can be treated like states and transitions while reducing the scalability and exploitation common to traditional state machines.

4.2 Academic Systems of Digital Emotions

Industry approaches to game AI can be applied to any kind of problem, from pathfinding to selecting optimal strategies to NPC behaviours. An algorithmic approach often has a narrow range of application because each algorithm is designed to tackle a single problem, and is often studied at length in academia. Simulating human behaviour, specifically emotional reactions to events, is a known problem with the goal of testing various psychological models to determine which is more likely to be the "correct" one. While they might be

unsuitable for video games, academic emotion systems can still be studied to learn new techniques and approaches that can either be used as-is or modified to meet a specific requirement that would be unrealistic in real life. Computer models of emotion are usually created as simulations to enable the testing of specific psychological models and theories. PARRY and BORIS are some of the first systems that attempted to simulate human emotions. More complex simulations include EMA, ACRES, the affective reasoner, and MAMID.

The first well-known computer model of emotions is PARRY, a program that simulates a patient with schizophrenic paranoid disorder (Colby, 1981). Built on the creator's theories of the condition, PARRY has an underlying structure of rules and tracks an internal emotional state. It is one of the first computer programs that passed the Turing test, even when its transcripts were presented to psychiatrists. Critics of PARRY have claimed that "errors" made by the system could be explained away due to the nature of the disorder being simulated. This could also be seen as a clever design selection knowing the limitations of computers at the time. Another early system is BORIS, a narrative comprehension program which answers questions about narratives that it has read to demonstrate its understanding of the narrative It accomplishes this by organizing causal relationships from the narrative into an internal representation of memory (Lehnert *et al.*, 1983). While BORIS is not an emotional agent simulator, it does comprehend the emotional state of characters in the narratives it analyses. It uses Memory Organization Packets (MOPs), a type discrete knowledge structure that might include goals, plans, and intentions, to describe conceptual dependencies, such as a lender-borrower scenario. MOPs can be linked to other MOPs or meta-MOPs depending on the current context. Thematic Affect Units (TAUs) and ACES are used to deal with reactions that are emotional in nature, which, combined with MOPS, allows BORRIS to infer what caused an emotional episode in a narrative. This organization of units dynamically forms episodes relevant to the current story that behave like static scripts. Despite being comparatively simple designs, both PARRY and BORIS demonstrate that the design space encompassing human behaviours and their accompanying psychological models and theories is too broad for a single system to handle. Therefore, assumptions must be made in order to narrow the scope of the design to ensure that the selected components fit the intended purpose.

As demonstrated with PARRY and BORIS, it is often required to make a series of assumptions to implement a psychological model in software since there is no complete, agreed upon model of human emotions. For that reason, each academic system tends to focus on a single aspect of the human emotion dynamic. The Emotion and Adaptation (EMA) framework was designed to explore emotion dynamics with the goal of explaining their processing structure (Marsella and Gratch, 2009). EMA specifically models the parallel

processes of appraisal dynamics, which are fast and automatic, and inference rules, which are slow and sequential, to simulate an agent's rapid series of responses to an event, its perceived consequences in relation to their goals, and its current understanding of the environment. It includes mechanisms for determining how much control the perceiving agent has over the outcome of an event, as well as causality and agency to estimate how other agents influence the outcome. The resulting configuration of appraisal values, called an appraisal frame, can then be matched to different, pre-designed emotion types. Appraisal frames contain propositional statements that are annotated with its relevance to the agent, desirability, likelihood, expectedness, causal attribution, controllability, changeability, and the perspective that the frame is viewed from. Perspective is included so agents can appraise an event from another agent's viewpoint. The ability to calculate the desirability and likelihood of an event to aid in the detection of future benefits and threats is enabled by decision-theoretic planning and a blackboard architecture. When an appraisal frame is created, a coping strategy simulated by a control signal is chosen. The chosen signal might depend on the controllability or ambiguity of the agent's internal world state representation at the time of appraisal. The control signal enables or suppresses cognitive processes, changing the agent's actions and appraisal dynamics. EMA cycles through the appraisal and coping stages during an unfinished event to update the agent's understanding of the world state, potentially passing through a number of appraisal frames before the event's conclusion. Mood is simulated by aggregating past appraisal frames and used to linearly augment the current appraisal frame by changing its intensity. This can influence the manifestation of the frame. While EMA presents a way of appraising and matching events to emotions, it also briefly touches on perspective and reasoning about how other agents are "feeling", which can consequently affect their own emotional processing.

The affective reasoner (Elliott, 1992) is a simulation-based system designed to test psychological models for use in language understanding programs, virtual tutors, and other similar platforms. It uses a combination of domain-independent rules for interpreting emotion eliciting events, a supervised learning case library for creating models of other agents' personalities, a world simulation based on object-domain theory, and a graphical user interface (GUI) for interacting with the system. There is one high-level domain rule for each emotion in the Ortony, Clore, and Collins (OCC) model of emotions (Ortony *et al.*, 1988), with which the system attempts to match a schema, which includes the agent's interpretation of an event, to an emotion template. The resulting template is used to determine how the agent reacts. Events are filtered by the agent's goals, standards, and preferences such that irrelevant events with respect to the agent are not processed. If an event is happening to another agent, an observing agent can try to reason about the affected

agent using the generated case library to determine their concerns and personality, while potentially eliciting an emotional response in themselves by proxy. The ability to "feel for" other agents enables rudimentary negotiation and opportunity recognition via reasoning about other agents' mental state. If multiple emotions are present, some are combined and actions present in both emotions are used to represent the compound emotion. In other cases, conflicting actions are arbitrarily removed. The determination of emotional intensity and duration are not mentioned in the affective reasoner design. The system's maintainability is limited because the addition of new events requires new emotion manifestations, personalities, simulation sets, and content theory, in addition to new interpretations and event handlers. Even with additional processing for determining how other agents "feel", each system still follows set algorithms and rules and some apparently irrational behaviours can still arise from these systems.

The Artificial Concern Realization System (ACRES) was designed as a test system for a partial emotion theory based on functionality (Frijda and Swagerman, 1987). Its goal is to show that the irrationality of emotion can arise from a functional system and implicit event appraisal is enough to stimulate action selection and execution. Unlike the other systems, ACRES does not have explicitly defined agent goals. Instead, it has a set of concerns that should be maintained from which goals can be derived when given information about the current operation environment. Emotions, therefore, are part of the concern realization process and are used for automatic detection of opportunities and threats, and controlling action priorities via interruption and persistent signalling. Concerns are represented as points against which actual data is matched. Multiple concern processing, including current environment evaluation, was ideally designed to work in parallel but could not be implemented in ACRES. The model behind ACRES proposes that an emotional system should have an action repertoire comprised of pre-programmed responses, including facial and verbal expressions, for dealing with limited information, time, or resources; general strategies that can be augmented with additional information; and planning capabilities for generating new action plans to deal with both planned and unplanned events. Actions are selected by first choosing a desired post-condition and matching the associated preconditions with the current situation. For competing actions, top priority is selected based on satisfaction criteria rather than maximising parameters. The strength of the relevance signal is determined by the associated goal priority and environment parameters. Social signals can be included for multi-agent environments by maintaining a record of past agent interactions for each relevant external agent. So far, the focus of the studied systems has been on either specific emotion persons or matching different appraisal results to emotion categories. However, personality is known to have an effect on how an individual appraises and manifests

emotions in response to events.

The Methodology for Analysis and Modelling of Individual Differences (MAMID) is a simulation-based, domain independent system that explores personality traits and their effects on the appraisal process and other cognitive functions, including attention and memory (Hudlicka, 2002). The system categorizes cognition, emotion, and personality separately to aid in parametrization so that individual factors can be applied to different processes and organizational structures. Multiple factors can be combined into a normalized, linear combination where each factor is assigned a weight based on its influence over the system. Emotions are represented both categorically as a state of appraisal and dimensionally as intensity and valence values. An agent's memory contains domain knowledge in addition to agent-specific beliefs, rules, and problem solving knowledge. Events, expected events, and goals are all modelled in the same manner because their main differences are the times at which they occur and their desirability. An interesting feature of this architecture is the offloading of emotional interpretations to the goal manager and action selection modules. This decision allows users to control how emotions manifest in the environment, including the appearance of compound emotions.

Since they were built to test emotion models or reason about emotion-inducing events, academic systems might not be suitable for video games due to their lack of domain focus, which increases their required resources and computation times. However, several lessons can be learned from these endeavours and applied to the design of the GLaDOS system. Many of these systems filter inputs based on goals, which can be used to improve overall performance. Each system also offers unique ideas including processing methods, reasoning about other agents, personality, and accepting apparently irrational behaviour from comparatively simple systems. PARRY demonstrates how the application context can be used to trick users into believing that it is an emotional being without using advanced AI techniques, while BORIS shows how traits from an event can be combined with conceptual dependencies and folk wisdom to infer a character's emotional state. The parallelization of fast and slow reasoning processes in EMA exemplifies how an agent can experience many different emotions during an event, and the affective reasoner illustrates how knowledge of other agents' emotional states can be inferred by observing their reactions. ACRES and MAMID have especially important information that can be useful when building an emotion framework for video games. ACRES proves that irrational emotional behaviour can still arise from a functional system, while MAMID shows how emotion, mood, and personality traits can be parametrized and applied to individual processes and data structures to control behaviour. Taken together, these academic systems illustrate how an underlying psychological model can be paired with domain knowledge and basic reasoning processes to create the illusion of emotional behaviour in

computerized agents. Although not directly applicable to video games, their successes can still be isolated and used in the GLaDOS system design space.

4.3 Systems for Generating Non-Player Character Emotions

Creating more interesting NPCs, agents specifically designed for games, is a task that has been approached from different angles in academia. For example, there has been a lot of interest in NPCs that learn such that player-tailored difficulty adjustment can be integrated into video games. Several academic AI techniques have inspired, or been applied to, these endeavours, including case-based reasoning (Molineaux *et al.*, 2005; Spronck, 2005; Sanchez-Pelegrin and Diaz-Agudo, 2005), reinforcement learning (Andrade *et al.*, 2005; Marthi *et al.*, 2005; Spronck *et al.*, 2004a), and evolutionary algorithms (Ponsen, 2004). Comparatively little attention has been given to making NPCs more believable, but the literature that has been released on the topic is promising. ALMA, GAMYGDALA, and the Em tools from the Tok agent architecture were created specifically to enable the creation of NPCs that react based on psychological models of emotion. However, all three systems use a combination of the OCC model of emotions and the Pleasure-Arousal-Dominance (PAD) space framework (Mehrabian, 1996) to create emotional variations, intensities, and sometimes moods. These models were consistently chosen because they are simple to understand compared to other psychological models, was designed for digitization, and is used by other video game researchers. Despite this commonality, each system explores a unique way of using the same model with different effects.

A Layered Model of Affect (ALMA) (Gebhard, 2005), in addition to implementing a version of the OCC model and PAD framework, simulates personality based on the Big Five psychological model (De Raad, 2000). Emotions, mood, and personality are treated as short, medium, and long-term system effects, respectively. Emotions are triggered by an external annotated event or object and decay quickly, but have a lasting effect on the system by altering the agent's mood once it is translated into PAD space. Personality is the "default mood" to which the agent returns in the absence of emotion eliciting events. ALMA has four sets of evaluation rules to account for differences in the cognitive processing of events, actions, and objects, external agent acts, and emotional and mood displays. Events, actions, and objects are annotated with appraisal flags to aid in their evaluation. The current mood is conveyed through animations, dialogue tones, and colour-coded dialogue text. When the system was tested, participants found the emotions more plausible than the moods (Bock, 2009), suggesting that short-term effects might be enough

to convey information about the agent's current state. ALMA is a great usage example for the implementation of the different aspects of an emotional episode, but it does not explain its portability and adaptation value between applications.

In contrast, the GAMYGDALA engine was designed to be modular, efficient, and psychologically grounded (Popescu *et al.*, 2014), similar to other game engines with specific purposes such as physics engines. GAMYGDALA is a black box design that provides game designers a way of introducing emotions into NPC groups such that it can be used in a variety of games and does not require any previous knowledge of psychology or appraisal theory. The engine assumes that a game can be represented as a series of events, and all NPCs have at least one goal. Sixteen OCC emotions were implemented, but only the desirability and consequences of an event are considered during processing. The intensity of an emotion is determined by how probable an event is to occur, which can be altered by new game events or information. Since they can be derived from OCC outputs, PAD values are used to supplement emotional values to create emotion variations. Personality is represented using default emotion and social states which, in the absence of goal-affecting events, are reached using decay functions from excited states. To support portability, the engine outputs values that can be used in any NPC component, including animations and AI modules, as opposed to directly controlling NPCs. This makes GAMYGDALA the only system to indirectly influence NPC behaviours, allowing it to be added to games more easily at any stage of development. Thus far, both ALMA and GAMYGDALA have proposed ways of programming NPCs to have emotions and personality. However, they are somewhat constrained in their usage because they retain control over some of the event processing aspects. This could potentially limit some character designs that do not follow the proposed evaluation structure offered by either system.

The Tok agent architecture is an Object-Oriented approach to creating believable agents that use artistic freedom as a core driving requirement behind individual modules, including emotions and social interactions (Reilly, 1996). Character personality is used to ensure that all the typically shallow capabilities of Tok work together to produce a cohesive set of behaviours, including actions, speech, movement, and goal maintenance. This can result in the illusion of complex, believable characters. A number of tools collectively named Emotion Architecture (Em) were designed to support the creation of believable emotional agents based on the belief that artists create believable agents, which might not be smart or realistic, while AI researchers create autonomous ones, which are built from scientific models. Inputs to Em can come from any component of the Tok architecture, including output from Em itself,

which are processed by emotion generators, implemented as IF-THEN structures dictating how a character reacts, including a type, intensity, cause, and an optional target. The default emotion generators are based on the OCC model and can be categorized based on which Tok structure they influence, such as cognitive-appraisal, body-feedback, and memory. The use of generators aids in processing precedence so that higher priority emotions are handled first. The generators produce emotion structures which are organized in hierarchies according to emotion class, which can be queried for values, targets, and "memorized" information such as past emotional structures. An overall class intensity can be calculated logarithmically from the structures in a hierarchy, which is used to determine general agent effects. The system allows for the exaggeration of some structures and the suppression of others, with the intensity of individual structures decaying over time. Once structures are present, a behaviour feature map is used to map structures to behaviours. This approach enables a fine grained control over how emotions are manifested in actions and allows emotion structures and behaviours to be designed independently. A text-based game was used to test the system to avoid the use of animations, a separate type of emotion expression. Of 17 participants, it was found that a character built with the Em tools was more emotional, had more personality, was more believable, and had better social interactions than one built without. Additionally, there was a positive correlation between emotion and personality. However, there were no conclusive results connecting suspension of disbelief with the Em characters. In some cases, response times were deemed too slow to be believable (1 – 2 seconds on average), and repetitive responses to player questions were seen as unnatural. There are several additional drawbacks to the Em tools, including flexibility, ease of use, code re-usability, and modularity. These results and drawbacks, while undesirable, are important for the GLaDOS system specification because they show how other factors can impact a system's acceptability that are not directly related to its functionality.

Compared to academic systems and NPCs that can learn, relatively little research has been conducted into emotional NPCs. ALMA, GAMYGDALA, and Em all use the same psychological model, but often the reasoning was more about convenience than design. Despite this, the work has been very promising. ALMA integrated personality and mood into its design, expressing emotions through actions and mood through animations and dialogue. Using personality as a default mood is a good idea to account for times when no emotion eliciting events occur. Through testing, emotions generated by ALMA were considered more convincing than moods, suggesting that emotions by themselves might be enough to create the illusion of believable NPCs. GAMYGDALA exemplifies how a complete implementation of a psychological model is not always necessary to create more believable NPCs. The engine

also demonstrates the plausibility of condensing an emotion architecture into an engine that can be applied to any game by treating the manifestation of behaviour as an external component. The Tok architecture is more like a collection of tools than a system, leaving the entire burden of implementation on the designer. While this negatively influences usability, modularity, and re-usability, the Em tools does highlight how personality must affect all aspects of NPCs, including emotional displays, in order to be believable. It is also a reminder that interesting characters are not necessarily realistic, so psychological models should be considered a tool rather than a necessity. This knowledge is used in the GLaDOS design to allow for greater flexibility in what options are available when the psychological theories do not provide enough information for a full system specification.

4.4 Conclusion

With the focus shifting to the creation of more engaging video games, it has been proposed in both academia and industry that the next major innovation will be realistic NPCs. This implies that new methods and tools need to be created since no combination of ad hoc methods, heuristics, or common algorithms has been used with complete success for this purpose. Success can be defined as a method that, when applied to NPCs, maintains player engagement by reacting to player actions and observable game world events in a believable manner.

Reactionary behaviour based on emotions is one possible method for creating this type of NPC realism. Several systems have been designed for academic exploration of emotion psychology and its effects on behaviour, but they might be unsuitable for video games. This is due to their extensive domain knowledge requirement, which can be difficult to specify and maintain. They do still demonstrate the importance of having an underlying psychological model to create diverse and interesting behaviours. These systems also consistently filter inputs to the system based on a set of goals or concerns. This idea can be transferred to other system specification, like the one for the GLaDOS system, to improve overall performance.

There are comparatively fewer systems designed specifically for believable NPCs in video games, but they draw on some of the same concepts as strictly academic systems. There is more variation in system designs and they usually focus on one element of emotionally-driven NPC behaviours to narrow their scope. ALMA focused on the expression of emotion through actions and dialogue, GAMYGDALA strove for an engine that could be used in a variety of games, and Em looked at the role of personality in believable character creation. A combination of these ideas into a single system would be a good next step to explore. Another concern to address is the lack of diversity in

psychological models used for video game applications considering that there does not appear to be any proof of finding and systematically discarding other viable psychological models. While proven to be flexible, the GLaDOS system will not use the OCC model as part of its underlying design to allow for a wider exploration of emotion theories and models. While the OCC model might prove to be superior in the end, it cannot be said for certain that this is the case now.

Chapter 5

The Psychology of Emotions

Psychologists have not yet agreed on a definitive theory of emotions, or even what an emotion is, but there are some common characteristics that many researchers agree on. Most researchers agree that there are rapid, automatic, and unconscious connections between emotion, cognition, and action. However, there is less agreement on the definition, structures, functions, and antecedents of emotion. Of these, the definition of emotion has the most variation between researchers, with some hesitating to propose a definition at all. Organized sets of responses, appraisal processes, action impulse, coping function are some commonly mentioned elements in otherwise unique definitions of emotion. Other common elements include physiological, subjective feeling, cognitive, and behavioural or expressive components. Less frequently included as part of their definition, researchers state that each emotion is unique in function, structure, and expression. This means that the GLaDOS and similar systems do not necessarily require an exact definition of emotion, but it does require a list of basic triggers, structures, and functions (Izard, 2010).

Emotions can be activated by innate and conditioned stimuli (Scherer, 2005), memories (Russell, 2003), appraisal processes (Kagan, 2007), social interactions (Hoffman, 2008), goal-related activities (Roseman and Elliot, 2008), emotion-cognition interaction cycles (Frijda, 1988), and spontaneous changes in neurobiological systems or processes (Lewis, 2005). In terms of computing, stimuli, memories, social interactions, and goals can be encoded into a database while emotion-cognition cycles can be represented with a feedback loop. While it might be beyond the scope of specific design requirements for the GLaDOS system, abstract neurobiological systems can be simulated as a separate module where their output is a neurological signal or chemicals.

Structures attributed to emotions include a neurological (Fox *et al.*, 2007) and response system (Kagan, 2007), feelings or a feeling state (Russell, 2003) with an associated cognitive appraisal (Clore and Ortony, 2008), and expressive behaviour (Levenson, 2003). The neurological and response systems are

like two components of a single architecture: one system gathers and processes environmental information while the other formulates responses. Cognitive appraisal is a component of the processing structure, the feeling state is the intermediary component that connects the processing and response systems, and expressive behaviour is the final output of the overall system.

Functions associated with emotions include event monitoring (Izard, 2009) and attention focus (Fredrickson *et al.*, 2008); process interruption (Cole *et al.*, 2008), cognition (Lewis, 2005) and action motivation (Mayer *et al.*, 2008); providing information to guide and coordinate engagement in the environment (LeDoux, 2008), including: the creation of a mental workspace to explore possible reactions or solutions to environmental events (Fredrickson *et al.*, 2008); response direction (Scherer, 2005); behaviour regulation (Campos *et al.*, 2004); and facilitating indirect communication via social signals (Griffiths, 2003; Plutchik, 2001). Monitoring can be compared directly with a sensory input component that gathers information from a system's surroundings, while focus would dictate what information the system filters from the raw input. Process interruption would refer to the interruption of current computations and behaviours in a computerized system which would activate cognitive processes to handle new data. The resulting state of values could be used to choose a set of related output behaviours, similar to the mental workspace that is formed from information provided by emotions. This concept could be expanded further by saving past emotional states so that they can influence the current computational process. Behaviour regulation and response direction are the processes that make the final decision as to which action in the workspace to take. Even though these functions result in actions that impact the environment, the selection of animations to play while the action is executed is guided by the indirect communication function, which showcases the "humanity" aspect of emotions that could truly affect player engagement.

Processes that regulate emotion include: spontaneous neurological processes, such as hormones (Ellsworth and Scherer, 2003; Goldsmith *et al.*, 2008); cognitive processes, which include social learning (Shweder *et al.*, 2008); developmental processes that formulate personality (Rothbart, 2011); and behavioural processes, including expression (Clore and Ortony, 2008) and action (Izard, 2009). Regulation processes are often considered to function differently for each emotion (Ellsworth and Scherer, 2003). With the addition of a database to store "memories", these concepts could be implemented on top of a basic emotional processing architecture to simulate learning how to control emotion processing and displays, as well as personality development. Even though these points, activation, structures, functions, and processes are generally agreed upon by psychologists and can be reasonably represented computationally, there is little consensus on how each of these factors work,

individually or together, or in what order they are handled. This makes it difficult to specify how any computational model would work beyond a general architecture specification.

Approaches to studying human emotions fall into three broad categories: physiological or neurological, experiential or cognitive, and expressive or evolutionary (Carlson and Hatfield, 1992; Keltner *et al.*, 2014). Physiological approaches, which include psychophysical and neurological approaches, associate physiological reactions to the onset of emotions as opposed to the elicitation of physical responses by emotions. Notable physiological theories include James-Lange (Lange and James, 1922), which states that emotions are derived from physical reactions, and Cannon-Bard (Cannon, 1929), which postulates that emotions and physical reactions occur simultaneously when the thalamus is aroused. Experiential approaches state that cognitive processes shape how people view events and their resulting emotional responses. Freud's psychoanalysis (Freud, 1965) and Jung's analytical psychology (Jung, 1966) are early examples of this approach, but it has more recently developed into a branch of cognitive psychology. Other theories in the experiential approach include Schachter-Singer theory (Schachter and Singer, 1962), which states that a physiological response occurs first and emotions occur as a result of labelling the cause, and cognitive appraisal (Lazarus *et al.*, 1980), which states that thought occurs when presented with a stimulus and physiological and emotional responses follow. The expressive approach focuses on the observable and measurable aspects of emotional behaviour and includes the traditional theories of Darwin's evolutionary theory (Darwin, 1872) and behaviourism (Watson, 1919; Skinner, 1965). In this approach, emotions serve a functional purpose in survival and adaptively evolve over time via genetics and social learning. More recent theories in this area include Plutchik's psychoevolutionary synthesis (Plutchik, 1980), which states that emotion, evolution, and behaviour are linked, and Ekman's theory of basic emotions (Ekman, 1999), which proposes that emotions are used to mobilize organisms to deal with interpersonal encounters quickly and that basic emotions are the same in every culture, but are expressed differently due to cultural learning. Since psychological theories of emotion have already been categorized by psychologists, analysing representatives from each class can help determine which groups are likely to contain workable theories that can be used in the GLaDOS design.

5.1 Choosing an Approach

Since psychologists cannot agree on processes or their order in the emotion cycle, choosing a theory to use in the GLaDOS system design requires a comparison between the three approaches and identified software requirements. The minimum creative requirement that is important at this stage is that the

chosen model of emotion convinces players that NPCs are emotional beings without harming their engagement with the game. Key functional requirements include minimizing the computational resources required to maintain the system, such as speed and memory, and modularity to aid ease of use. These become increasingly important with game scale, where there can be several dozen NPCs in any given level.

It is unlikely that physiological approaches are suited for this task. Consider the conflicting theories of James-Lange and Cannon-Bard. The James-Lange theory (Lange and James, 1922) states that emotional arousal follows from a physical reaction and visceral arousal. If this theory is implemented in a game, physical reactions to events would likely be no different than current static scripting processes, as these behaviours occur before emotional processing. This would imply that an NPC's behaviour would occur as a result of an animation – the closest thing an NPC has to visceral arousal. The Cannon-Bard theory (Cannon, 1929) states that emotional and physical arousal is a result of arousing the thalamus, a structure in the brain that processes and relays sensory information to other brain structures. While this theory is more feasible to implement because animations and behaviour occur in parallel, the human brain is a complex structure that is not well understood. Modelling a brain for use in a video game would likely use more development time and computational resources than can be afforded with no guarantee that the model is correct.

Experiential approaches, which encompass motivational and cognitive theories of emotion, are a better candidate. The Schachter-Singer theory, which draws on the physiological theories of James-Lange and Cannon-Bard, proposes that a stimulus causes a physiological reaction, but an emotion does not occur until the individual has identified a reason for the reaction (Schachter and Singer, 1962). While this theory has the same implementation concern as the physiological theories, it does propose some useful elements: emotions arise from an appraisal of a stimulus, different emotions can be derived from the same stimulus depending on its interpretation, and physiological arousal is essential to the production of emotion. Cognitive appraisal (Lazarus *et al.*, 1980; Lazarus and Lazarus, 1996), proposed by Richard Lazarus, is another theory that relies on the evaluation of a stimulus to produce an emotion class[1]. This theory is a better candidate for the GLaDOS design because it proposes that appraisal leads emotional and physiological responses. This can be compared directly to treating game events as appraisal inputs and producing behavioural changes and animations. The theory also proposes an evaluation cycle where a primary appraisal stage evaluates what the event is, a secondary appraisal

[1]Identified emotion classes include: Anger, Anxiety, Fright, Guilt, Shame, Sadness, Envy, Jealousy, Disgust, Happiness, Pride, Relief, Hope, Love, Gratitude, Compassion, Aesthetic experiences (Lazarus, 1991)

stage determines what needs to be done in response, and a reappraisal stage observes the impact of the executed behaviour to determine what to do about the altered environment. This type of cycle could proceed indefinitely, so a cut-off point would need to be defined for persistent stimuli. However, while this theory proposes a process that could be implemented in computers, some of the emotions and methods of accessing them are relatively complex. Complex emotions might not be required for all NPCs in a game specification and forcing their inclusion could increase resource demands on both the human designer and the host game. This could adversely affect player engagement if the extra strain creates unexpected response delays or affects the underlying functionality or causes a game designer to underspecify their game's design domain.

The first type of emotion theory proposed in science, expressive approaches, is based on Darwin's evolutionary theory which proposes that emotions play a central role in a species' survival (Darwin, 1872). Robert Plutchik's psychoevolutionary synthesis (Plutchik, 1980) proposes that emotions help organisms deal with survival issues present in their environment. It also states that there are eight prototype patterns, or basic emotions, organised into pairs such that paired emotions cannot be expressed simultaneously. In this theory, complex emotions are derived from the basic emotions using traits such as emotion intensity or by combing multiple categories into a single definition. The concept of a combining a discrete number of patterns and traits into an unknown number of complex ones supports the ability to specify a subset of emotions to be implemented on all NPCs which can also be extended to create complex computational behaviours based on an NPC's purpose. Grouping emotions into pairs with variable intensities could also be helpful when deciding how emotions can be represented numerically in the GLaDOS system. Although this theory clearly defines basic goals and a finite number of basic emotions that can arise from those goals, the proposed framework for determining how emotions are derived from stimuli appears to be a looser description of cognitive appraisal (Plutchik, 2001). This would make it difficult to completely specify the GLaDOS design using this theory alone. Another expressive approach, Ekman's theory of basic emotions, states that basic emotions exist to deal with fundamental life tasks and that emotional expression developed as a means to communicate the expresser's internal state. Basic emotions, a label for a family of related emotions, can be distinguished by universal signals and physiological reactions. The universality of signals in different cultures is attributed to convention rather than evolution, but differences in the amplification of emotional signals and manifestations are a product of the culture's social learning (Ekman, 1992). This observation can be used to explain manipulative behaviour, where an organism will feign an emotional expression to elicit a response from others (Ekman and O'Sullivan, 1991). Ekman

also proposes an appraisal mechanism similar to cognitive appraisal, but it distinguishes itself by stating that the appraisal is quick, non-reflective, and unconscious, and it can be influenced by social learning, personal history, and personality (Ekman, 1999). This theory can potentially be used in a non-essential GLaDOS system component which can be used to change how an event is interpreted by an NPC based on their in-game culture. However, the number of basic emotions encompassed by this theory has not yet been confirmed, making it difficult to specify implementation requirements for the GLaDOS system.

Choosing a theory to derive GLaDOS system specifications from should be guided by a comparison of the emotion theory categories based on core system requirements, such as maintaining player engagement, conserving computational resources, and modularity. These requirements can be used to immediately disqualify physiological approaches because they state that emotion follows physical or neurological arousal. This implies that a model of the human brain would be necessary to coordinate stimuli, behaviours, and animations. Modelling a brain would likely require more computational resources than necessary for the intended purpose, with additional development time and no indication that player engagement might be enhanced. Experiential approaches, such as cognitive appraisal, propose a process of deriving emotional states from evaluations of environmental events and stimuli which can be easily compared to computational processes. However, the types of emotions and definitions associated with these theories often contain complex and difficult to understand members, such as distinguishing between envy and jealousy, making them more difficult to specify computationally. In some cases, not all the emotions are necessary, and computing their values wastes computational resources. In contrast, expressive theories, such as psycho-evolutionary synthesis, are less specific about how emotions are formed but more succinct in their definition of what emotions are essential and how they can be computed. For video games, this implies that a good system would combine two or more theories to achieve a balance between engagement, resource consumption, and modularity.

5.2 A Combination of Theories for Video Games

A basic architecture for simulating emotions is comprised of three parts – process, emotion state, and behaviour expression (Campos *et al.*, 2004; Sloman *et al.*, 2005). The process structure is responsible for monitoring the environment and filtering incoming information, comparable to attention focusing, based on a set of goals such as self-preservation. After identifying relevant

ones, stimuli need to be transformed into a set of emotion values. The transformation process can be affected by past emotion values, personality, and relevant goals. Once emotion values have been stored, they can be referenced by the behaviour expression component that selects an action to take, an accompanying animation for the social aspect of emotional expression, and a target that the response is directed towards. Part of the action selection process is identifying actions that are not appropriate for the current emotion state and omitting them, effectively narrowing the selection space. An additional function, process interruption, can be triggered at any stage, but it is always used when a new behaviour is chosen to ensure that it is executed. If a more advanced architecture is required, memories, images, relationships, cultural quirks, and some neurobiological systems such as hormones, can be included to augment the filtering and transformation processes. Learning algorithms could also be used in a more advanced system to change the available actions and animations in the behaviour expression component. However, the proposed basic architecture does not adequately specify how the transformation process works, or what emotional values are produced. These two areas require more specific psychological theories and their specification is what defines the GLaDOS system.

Cognitive appraisal describes a process of translating external events into emotion values by determining how those events change the status of a goal or other internal beliefs (Keltner *et al.*, 2014; Lazarus *et al.*, 1980; Oatley and Johnson-Laird, 1987). There are two appraisal stages – primary and secondary – followed by a coping strategy. In primary appraisal, the relevance of an environmental event in relation to goals and ego[2] is determined. If the event has no relevance to any held goal or ego, there is no change of emotion. If the event is relevant, the congruence or positivity of the affected goal or ego is calculated. The result of this calculation can be used to immediately discount a number of possible emotions based on its desirability, where a positive emotion is more desirable. Secondary appraisal is the process of determining what responsibility to assign the source of the stimuli, if an action can be taken, and what the future consequences of the reacting to stimuli could be. After emotion effects are calculated, a coping strategy is created that includes the selection of an action and emotional display in response to the stimuli (Lazarus, 2006). In terms of the GLaDOS architecture, primary appraisal occurs in the process component where emotion values are calculated, while secondary appraisal and coping occurs in the behaviour expression component, where an action and social expression are chosen and executed. While the appraisal process proposed by Lazarus and others often have a number of accompanying core relational themes for different emotions, there is variation in the number of

[2]Some components of ego include social and self-esteem, moral values, ego ideals, meanings and ideas, other people and their well-being, life goals (Lazarus, 1991)

emotions and whether the emotions are discrete classes (Plutchik, 2001; Ortony *et al.*, 1988; Oatley and Johnson-Laird, 1987) or the result of a combination of continuous dimensions such as arousal and pleasantnesses (Ellsworth and Scherer, 2003; Fontaine *et al.*, 2007; Mehrabian, 1996). This suggests that the appraisal process itself is agnostic to the emotions implemented, allowing the freedom to select which emotions to include and how. Unfortunately, this also makes it impossible to completely specify how the appraisal process works in the GLaDOS design, leaving the remaining element definitions to individual implementations.

To enable the creation of a variety of NPC behaviours, it is potentially beneficial to encode a select number of basic emotions into the GLaDOS design which can be combined to form more complex emotions if required. This ensures that all NPCs have the same set of core emotions, but that "special" NPCs can be given additional emotion definitions as required. Plutchik's psycho-evolutionary synthesis, which refers to a series of processes similar to cognitive appraisal (Plutchik, 2001), proposes a taxonomy with eight basic emotions[3] and a structural model for synthesising complex emotions from basic emotions and their intensities. For example, *Delight* is defined as a combination of *Joy* and *Surprise*, whereas *Contempt* is a combination of *Anger* and *Disgust*. The basic emotions are organized into pairs where a positively aligned emotion is matched with a negatively aligned one (Figure 5.1). This can be used to balance emotion values in a computerized system, and should be applied to the emotion state component of the basic GLaDOS architecture in two parts. The mandatory component contains the eight basic emotions, their current values, and decay functions that cause the values to return to an equilibrium state in the absence of relevant environmental events. The optional component is a combinator that is used to encode complex emotions as they are required.

A complementary theory that can be applied to the emotion decay functions is the Opponent-Process theory proposed by Richard Solomon (Solomon, 1980). This theory predicts that an aroused emotion will peak and decay to a steady level, and then over-adapt to a smaller peak in another emotion before returning to an equilibrium state. This can be applied to emotions that exceed a high arousal threshold to produce effects such as relief after a state of terror. Which emotions follow from another can be encoded as emotion combinator rules in the cases where the selected emotions are not pairs in Plutchik's taxonomy. Emotions that are pairs in the taxonomy could be encoded as functions in the emotion decay specification.

The three units in a basic, computerized emotion architecture – process, emotion state, and behaviour expression – contain specifiable components,

[3]Joy, Sadness, Trust, Disgust, Fear, Anger, Surprise, and Anticipation

Figure 5.1: Plutchik's Emotion Solid with eight basic emotions, three intensity levels, and eight secondary emotions

such as sensory input module and filtering, which can be drawn from commonalities in proposed emotion theories. It is less clear which algorithms are required to transform events into emotion values and which emotion categories are required to define an emotion state due to the variety of research perspectives on emotions. The decision to use Richard Lazarus's cognitive appraisal and Robert Plutchik's psycho-evolutionary synthesis to guide the specification of these unclear areas are what elevate the general emotion architecture to the GLaDOS system. Cognitive appraisal is a good candidate for the transformation process because it evaluates an event, in relation to goals, in two distinct evaluation stages which can easily be translated into algorithms. This theory also extends into the behaviour expression unit of the architecture by specifying that emotional arousal should result in an expression of the emotion. The number of emotion values produced should ideally be flexible to allow the addition of more emotions to the GLaDOS design as they are required to enable the creation of more complex behaviours. This is supported by psycho-evolutionary synthesis, an emotion taxonomy and structural organization based on eight basic emotions. The proposed basic emotions are arranged into opposing pairs, which enables the rebalancing of values caused

by new emotion values or emotion decay to an equilibrium state in a computerized system. The decay functions implemented in the emotion state can be complemented by Richard Solomon's Opponent-Process theory, which causes another emotion value to spike after experiencing an extreme emotional reaction. Additional decay rules can be implemented in cases where the opposing emotion proposed by Plutchik does not match the desired NPC behaviour. With these psychological theories as the foundation, a GLaDOS system design that is well-founded, resource-friendly, extensible, and modular can be specified.

5.3 Conclusion

Even though a general architecture can be agreed on by many researchers, the exact order and method used by emotion processes has yet to be specified. As a result, three broad classes of emotion theories have emerged. Physiological theories state that emotions follow from neurological or physical arousal; experiential approaches propose that emotions are a product of environment and event appraisals; and expressive approaches suggest that emotions developed evolutionarily as a method for dealing with environmental stimuli to ensure survival. In order to choose an approach, and subsequently a theory of emotion to follow, a basic set of GLaDOS system requirements was formulated, which includes maintaining player engagement, minimizing computational resources, and component modularity. While physiological theories can be immediately omitted, there is less certainty about which of the remaining approaches, experiential or expressive, is the ideal candidate. Since experiential theories tend to have more defined processes, whereas expressive approaches tend to have more succinct collections of emotions and descriptions, a hybrid of the two might be the best choice to achieve the goals of engagement, resource management, and modularity.

A basic emotion architecture can be formulated from general principles of emotion research which contains three main units. The process unit is responsible for taking environmental events and transforming them into emotion values; the behaviour expression unit takes emotion values and determines what behaviour and animations should be executed; and the emotion state, which stores emotion values and returns them to equilibrium values via decay functions. While these common elements can be used to define many areas of an emotion architecture, there are still areas that are left undefined. The specific selection of emotion theories is what differentiates the GLaDOS system from a general design. Richard Lazarus's cognitive appraisal, an experiential theory, can be used to transform environmental events into emotion values based on their relation to goals and ego, with room to include additional factors such as social learning and memories. This theory also extends into the behaviour

expression unit by helping to determine what action to take in response, where the action should be directed, and what other emotional expressions should be used. The most important, unclear, specification in the emotion state is how many and which emotions it should include. Robert Plutchik's psycho-evolutionary synthesis, an expressive approach, proposes a taxonomy of eight basic emotions and a structural organization that allows them to be combined into additional, complex, emotions. This allows for the ability to add more emotions if more complex behaviour is required. The basic emotions are also arranged into opposing pairs, which enables the rebalancing of emotion values if there is a change, or the emotions are decaying to equilibrium. An additional theory, Richard Solomon's Opponent-Process, can be used to complement the emotion decay functions by simulating follow-up emotion values after extreme emotional states. This suggests that the emotion state has three subcomponents – an emotion unit where basic emotion values are stored, a combinator layer where emotional values are combined to determine if a complex emotion is being experienced, and decay functions. Even though it would be simpler to select a single theory for the GLaDOS specification, the integration of cognitive appraisal, psycho-evolutionary synthesis, and Opponent-Process theory could reduce the number of unspecifiable system elements, produce the desired results, and be more flexible than if one were used alone.

Chapter 6

Designing the GLaDOS System

The GLaDOS system is designed for game designers and programmers to enable the creation of more realistic NPC reactions to player actions and in-game events. The end goal of this effort is to enhance player engagement and maintain long-term interest in the games that implement this system. Since player engagement is currently a subjective measurement, the GLaDOS system must be tested with real players to gauge its effectiveness. One possible test is to create a small game environment, or an enclosed level in a pre-existing game, and allow players to interact with the game for a set amount of time. Once the time is up, the player is asked if they know how much time has passed. If their guess is smaller than the time slot, it is likely that the game is engaging. If the guess is close to or greater than the elapsed time, it is unlikely that the game is engaging.

For the initial GLaDOS design, only CRPGs are considered because one of their main concepts is to allow the player to create and play out a role, and their engagement can be negatively affected if the game world does not help them to maintain their chosen image. A highly visible aspect of a game is NPC behaviour and an NPC's ability to react differently to the player can heavily influence player engagement and interest. Even if certain NPC reactions cause detrimental effects to the player, it might still be enjoyable because it maintains the player's image of themselves inside the game (Yannakakis and Hallam, 2005). The initial GLaDOS design built using the psychological theories of cognitive appraisal for event evaluation, and psycho-evolutionary synthesis for defining emotions. This constraint is mandatory due to the volume of emotion theories which cannot be represented simultaneously due to conflicts. Since NPC animations are a key factor in conveying emotions, this system shall be built as a game mod, which allows the use of pre-existing game assets and environments. This limitation is also used as a partial test of portability by determining how much of the resulting GLaDOS system needs to be game-dependent. This constraint also limits the development platform to PCs

because game consoles do not generally support user-generated content.

At this time, Bethesda's *The Elder Scrolls* series has a large and active modding community, making it an ideal candidate for implementation and testing. It is assumed that the animations available in *The Elder Scrolls V: Skyrim* are sufficient for conveying basic emotions in both body language and facial expressions. It is also assumed that the game environment that the system is tested in has a limited number of NPCs so that even suboptimal processes can still produce acceptable execution times. For testing purposes, the most recent entry in the series, *The Elder Scrolls V: Skyrim*, supports keyboard and mouse, Xbox 360, and Xbox One input devices, allowing players to select their preferred input device, which can indirectly affect their engagement with the game.

6.1 Use Cases

The purpose of the GLaDOS architecture is to augment NPC behaviours such that they react in believable and interesting ways to in-game events and the player. This behaviour is important to both the player and the game designer, since they are both concerned with player engagement and entertainment. What qualifies as believable and interesting behaviour depends on the designer, who is knowledgeable about the game world, and the player's perception of the choices that the designer has made. Despite this, a good architecture design should only require game specific information from the designer, such as event and goal definition, process modifiers, and final behaviour expressions.

A basic system built around Lazarus' cognitive appraisal and Plutchik's psycho-evolutionary synthesis requires input and output interfaces, an event processing unit, and a persistent emotion state. In addition to defining the interfaces between the GLaDOS architecture and the game world, a game designer will most likely want to either modify existing modules or add a new module of their design. The use cases aimed at game designers focus on modifying and augmenting the basic emotion architecture that is already defined.

These design expectations and constraints resulted in a set of nine use cases (Appendix B.2.5). The first, and most important, use case is concerned with the GLaDOS system's ability to cause NPCs to react to events that have been identified as emotion-inducing. This scenario requires the basic system to accept and process event identifiers using cognitive appraisal, translate them into one or more of the psycho-evolutionary synthesis emotions, and select an appropriate behaviour to express the resulting emotion state. Two use cases were formed to capture the need to connect the system and the game. Unless an event or object is identified as emotion-inducing before it is encountered, the GLaDOS system is unable to recognize and process it leading to

the usage scenario of defining an emotion-inducing event as part of the design specification. This usage scenario describes the process of identifying key characteristics and adding an identifier and categorization, such as ("Enemy", "–HP"). This process allows game designers to tag events that they think should be processed by the system and categorize them in a way that suits their game. At the other end of the GLaDOS system, game designers need to be able to designate the type of NPC behaviour that they expect from different emotion states, including actions and animations. This requires designers to choose which emotion states trigger different reactions.

The next several use cases are concerned with customizing the basic GLaDOS design. There are two types of customization – modifying existing system components and augmenting the system with new components. Goals are the main cause of emotion states, as they define which events are relevant and how they are evaluated. While basic goals, such as survival, are already included, it is expected that designers want to be able to create new goals for their NPCs, such as having more gold than the player. This leads to the scenario where the designer must represent the goal in measurable terms and assign it a priority so that when multiple events occur simultaneously, events that affect high priority goals are processed first. Another system customization that designers might expect is the ability to define their own emotion categories if the system's basic emotions[1] do not capture the emotion state that they require. For example, if a designer wants different types of behaviour for "Anger" and "Rage", they would define at what emotional intensity "Anger" becomes "Rage" and connect their behaviour expressions to each definition.

Part of the human emotion process is the decay of emotions to a resting state in the absence of personally relevant events, a characteristic that designers might want to simulate or exaggerate. The resulting use case of defining emotion decay functions and emotional equilibriums emerges from this observation. Another factor that many people consider when describing people's emotional reactions, and that designers might want to capture, is their personality, leading to the use case that describes the definition of personality traits as processing modifiers. The choice to target the process modules of the GLaDOS system instead of the behaviour expression is a result of the definition of reactionary behaviour and complex emotion use cases, which already controls how emotions are expressed. The final use case that is concerned with adding user-defined modules is included to capture all scenarios where a game designer needs to add functionality that does not quite fit in pre-existing modules. This can be viewed as a generalized use case for the more specific case for defining personality traits. For example, if a designer has different cultures in their game, they might want to add a new unit that defines different social norms and customs. Depending on how that culture operates, this module

[1] Joy, Sadness, Disgust, Trust, Anger, Fear, Anticipation, and Surprise

could alter any number of basic functions, such as attention, appraisal, or behaviour expression. Since it is unknown at this time which module would be best to connect this new unit to, it is best to make it as easy as possible to add new modules to any connection point in the basic GLaDOS system.

Even if a game designer is able to customize and augment the basic system, the development process is useless unless they are able to install and test it. This leads to the final use case, which states that the GLaDOS system must be installable as a *The Elder Scrolls V: Skyrim* mod. This directly relates to the development constraint requiring the use of using this game for testing purposes.

These nine use cases are designed to capture the foreseeable expectations of players and game designers. While the player is unconcerned with the internal system functionality, they still expect the GLaDOS system to produce believable and timely NPC reactions to in-game events. They also expect to be able to install the system in a familiar way, which would be as a mod for *The Elder Scrolls V: Skyrim* in this case. In addition to player expectations, game designers have several expectations regarding the modification and augmentation of the base system to suit their needs. Essential GLaDOS system modifications include defining emotion-inducing events, what NPC behaviours arise from each emotion state, and what goals an NPC should have. More complex modifications that a designer might want to make include defining additional emotions such as rage, controlling the rate at which NPCs return to an equilibrium emotion state, and adding personality traits that alter how events are processed. In cases where the desired effect cannot be achieved by modifying existing system modules, a game designer might want to create and integrate their own system module. As with any design, these usage scenarios lead to a number of requirements that should be satisfied in the resulting GLaDOS system design.

6.2 Requirements

Since one goal of the GLaDOS system is to be customizable for any game that uses it, the product requirements focus on the functionality (Appendix B.3) required to implement a basic model of the psychological theories of cognitive appraisal and psycho-evolutionary synthesis. Additionally, there is an emphasis on performance-related, non-functional requirements (Appendix B.4) because a resource heavy system can have detrimental effects on the key task of player enjoyment.

6.2.1 Functional

A basic model for cognitive appraisal requires event monitoring, a prioritized list of NPC goals, primary and secondary appraisal stages, and mechanisms for executing emotion-induced behaviours. Event monitoring is necessary to acquire information from the environment, goals are required for extracting relevant events from incoming information, and events are converted into emotion values during the primary appraisal process. The goals must be prioritized so that a related requirement, process interruption, can be used. Process interruption ensures that events that affect critical goals, such as survival, are processed before others. This can help NPC behaviours appear more believable in an environment where multiple events occur simultaneously. Process interruption is also used when the secondary appraisal stage finds that the current NPC behaviours do not match the current emotion state and selects a new set of scripts and animations, collectively called a behaviour class. Scripted behaviours and animations are listed as two separate functional requirements such that they can be triggered separately, allowing greater control over how NPCs react to different emotion states. For example, an emotional change might occur such that a manipulative state is triggered, requiring mismatched displays of action and body language.

Modelling the taxonomy of emotions in psycho-evolutionary synthesis relies on the maintenance of a data store containing a value for each of the eight basic emotions, and methods for updating, rebalancing, and decaying these values over time to a specified equilibrium. Since the basic emotions in this theory are arranged in opposing pairs, rebalancing is required to ensure that each pair of emotions remains in a specified equilibrium. While emotion decay is not a core feature in psycho-evolutionary synthesis, it is required to ensure that changes to the emotion state do not persist indefinitely after an event has passed. Another key aspect of psycho-evolutionary synthesis is the ability to define complex emotions as combinations of basic emotions and emotion intensities. This enables game designers to specify more complex behaviours than the basic emotions can afford. This requires the GLaDOS system to be able to refer to and recognize any user-defined emotions that might be added.

In addition to modelling psychological theories, there are other functional requirements necessary for this type of system. For the system to be self-contained, a list of event identifiers and traits must be maintained. Recognizing processing modifiers and interfacing with user-defined modules is required so that designers can alter or elaborate on how events are processed, enabling the simulation of various personality traits and neurological systems. Most importantly, the ability to install the resulting GLaDOS system implementation as a mod for *The Elder Scrolls V: Skyrim* is necessary so that it can be tested for player engagement and entertainment.

6.2.2 Non-Functional

While several of the functional requirements have focused on internal system functionality, which the player is generally unconcerned with, the non-functional requirements have an impact on both the game designer and the player. Some types of non-functional requirement types, such as safety critical and security requirements, are not necessary due to the scope and nature of the GLaDOS system. There are also some requirements, such as adhering to anti-plagiarism rules, being aware of copyrights, and addressing other stakeholder standards, which do not concern the designer or the player. However, they are still necessary in order to fully define the scope of the design constraints. Ideally, the GLaDOS system will eventually be released to the *The Elder Scrolls V: Skyrim* modding community for feedback to see if a wider audience of players enjoy it. The specific target game limits what implementation language to use as well because *The Elder Scrolls V: Skyrim* is written in a scripting language called Papyrus, which cannot be interfaced with other programming or scripting languages. It is important to note that, although they might not be directly influenced by them, game designers are still concerned with player-oriented requirements since the player heavily influences the designer's work.

Arguably the most important group of non-functional requirements is related to system performance because it can affect what a game designer can do with the GLaDOS system and impact a player's engagement. This resulted in several quantifiable requirements, including how quickly the system must respond to in-game events, how often the system polls the game environment for new events, how many events are held at one time, prioritizing which events to process in heavy-load environments, and how many other GLaDOS systems might be running simultaneously. If the system is too slow, NPC reactions will be out of sync with their trigger, which might look strange depending on the context. It is also common in CRPGs for there to be more than one NPC in any given area, emphasising the importance of system efficiency. Other performance requirements that are concerned with the player's perception of NPCs include reliably returning an NPC's emotion state to an equilibrium state in the absence of other events, and running the GLaDOS system as long as the associated NPC is present in the game environment. Miscellaneous requirements that are more of a concern to the player are the adherence of events, goals, and behaviours to established game world lore and common knowledge of human behaviour, and the ability to install the system as a mod for *The Elder Scrolls V: Skyrim*, a common method for user-created game content.

The game designer also has several additional non-functional requirements, mainly concerned with how the GLaDOS system is built. It can be assumed that many game developers have at least some knowledge of Object-Oriented programming. Therefore, the product must adhere to standard

Object-Oriented programming practices so that it can be easily understood for modifications and extensions, such as accepting user-defined event classifications. This also leads to the requirement of system organization so that modifications can be easily made. The purpose for ensuring an organized system is to aid a designer's ability to comfortably use the GLaDOS system within a set time period. Another common characteristic of game designers to consider is that they usually have little or no knowledge of psychology, making it important to avoid the use of psychology jargon in system descriptions. The maintenance of the GLaDOS system will most likely be left to the users, requiring that essential usage information be integrated into the system's code and the creation of a companion usage manual for non-essential information. Should the GLaDOS system prove successful in improving player engagement and entertainment, it might be ported to other languages for use in other games. Therefore, the system also needs to be written in a way that aids in this process.

While the functional requirements specify different processing stages and what is expected from the GLaDOS system, the non-functional requirements are what define a successful design within the context of a game. If the performance requirements are not met, the system will be sluggish and appear to be broken, while many of the game designer specific requirements specify how easy it is to use and customize. This implies that there is more work in the organization and efficiency of the GLaDOS system than there is in adherence to the psychological theories behind it.

6.3 System Architecture

The basic GLaDOS architecture is comprised of three core units (Figure 6.1). The Process Unit gathers and processes information from the game environment; the Emotion State contains the current emotion values of the NPC and associated balancing and decaying functions; and Behaviour Expression determines what actions and animations to produce from emotion values. While many of the components are generalized and widely supported by psychologists, such as sensory input and attention, Richard Lazarus's cognitive appraisal is used to specify how game-world events are converted into emotion values and Robert Plutchik's psycho-evolutionary synthesis specifies what emotions can be produced and how they are stored. While cognitive appraisal does propose a set of core emotional themes, psycho-evolutionary synthesis proposes eight basic emotions, with the ability to specify additional, more complex, emotions. This gives designers flexibility when creating NPC behaviours by providing a framework for defining a more finely grained specification of when and how certain behaviours are triggered. Conversely, psycho-evolutionary synthesis proposes a method of processing stimuli that is similar

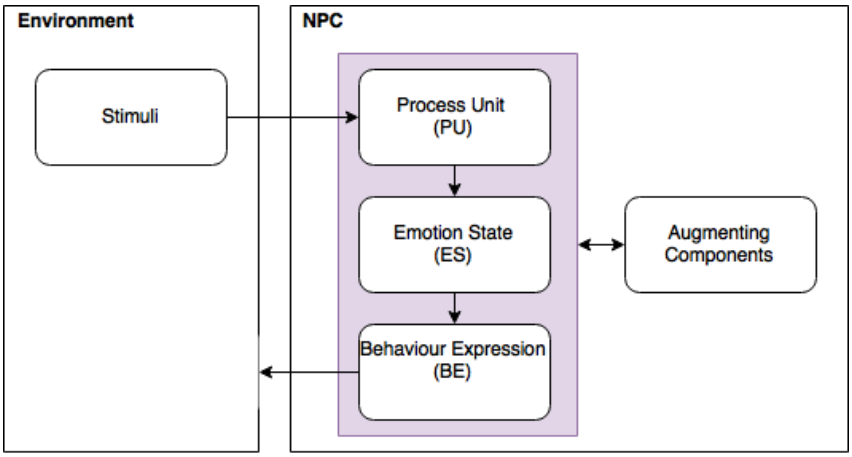


Figure 6.1: Main Architecture Units in the GLaDOS Design

to cognitive appraisal, but with fewer details. This means that it is possible to use cognitive appraisal instead, with similar results, while enabling the GLaDOS architecture to have more specific process definitions. Cognitive appraisal has three processing stages: primary appraisal calculates how an event is relevant to an NPC; secondary appraisal assigns responsibility, determining if anything can be done about the event, and identifying future consequences; and coping decides what, if any, behaviour and emotional display, such as facial expressions and audible cries, to trigger. Since it is possible for the results of secondary appraisal to change the current emotion values, the results of this stage could be sent to the Emotion State as an additional input in a more complex design.

6.3.1 Process Unit

The Process Unit (Figure 6.2) is responsible for gathering and filtering goal-relevant information from the game environment and determining what emotional impact that information has on the NPC. This unit has four components: Goals (PU-1), Sensory Input (PU-2), Attention (PU-3), and Primary Appraisal (PU-4).

PU-1. **Goals** (Functional Requirement F–2)
The data store associated with the Process Unit, Goals, is key to ensuring that only relevant information is processed inside the GLaDOS system. This module contains all information required to be able to associate game events with NPC goals and transform those events into emotion values.

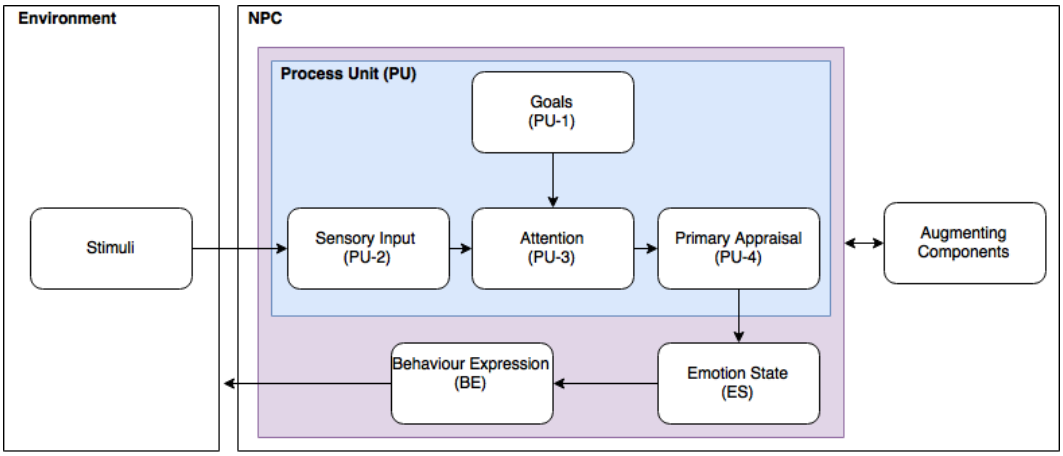


Figure 6.2: Process Unit Components

PU-2. **Sensory Input** (Functional Requirement F–1)
The Sensory Input module is a proxy layer between the game environment and the GLaDOS system, gathering game events and converting them into system-friendly data structures. It might only need to be defined once per game since all NPCs can use the same module because it does not affect how data is processed internally. The data structure created by this module must include an event source for assigning responsibility, and identifying event characteristics to help match it with a Goal from the Goals module (PU-1). For example, if a dragon landed nearby, the event source could be "Dragon" or "Opponent" and one possible characteristic is "DamageHealth" signifying a potential loss of health points. Due to the complexity of CRPGs, a significant amount of information can be queued for processing, requiring this module to be computationally fast and efficient to prevent delayed responses. This is due to a lack of guarantee that any information parsed in this module is relevant to the NPC.

PU-3. **Attention** (Functional Requirement F–3, F–5)
The Attention module is responsible for filtering information parsed by the Sensory Input module (PU-2) based on their relevance to goals found in the Goals module (PU-1). If an information unit does not relate to any goal, it is discarded. Otherwise, a reference to the affected goal is added to the information unit and passed onto the Primary Appraisal module (PU-4). Goal relevance is handled very generally in this module, as a more thorough analysis is performed in the Primary Appraisal module where the emotion values are derived. If an information unit affects a high priority goal, this module has the ability to interrupt the Primary

Appraisal module to ensure that this information unit is handled immediately. This ability is crucial in situations such as "fight-or-flight", where an NPC should stop all activities and get to a safe area.

PU-4. **Primary Appraisal** (Functional Requirement F–4)
After environmental events have been converted into data structures and filtered by goal relevance, the information is processed in the Primary Appraisal module, named for the first component of Lazarus' cognitive appraisal theory (Lazarus *et al.*, 1980) where an event's relevance to goals is determined. In a computational context, this means processing the event data and the referenced goal to determine how it affects the NPC. The result of this appraisal is passed to the Emotion Store module in the Emotion State (Section 6.3.2).

The purpose of the Process Unit is to determine which game events are relevant to the NPC's goals and create a package of information that can be used by other GLaDOS system modules to generate a response. Only events that affect NPC goals, which cause emotional changes, pass through to the Emotion State, where emotion values are stored and maintained. The importance and emotional impact of NPC goals is contained in the Goals module (PU-1), a collection of values that can be easily understood by the computer. Raw environmental data is gathered by the Sensory Input module (PU-2), which parses it into information packets that the system can understand. This supports portability and flexibility because it does not require the GLaDOS system to be rewritten for similar games. Completed information packets are passed to the Attention module (PU-3), where they are checked against goals for relevance. If a packet is not relevant to any goal, it is discarded. If a packet is relevant, a reference to the goal is appended and the packet is passed to Primary Appraisal (PU-4) for processing. In the Primary Appraisal module, event information and goal statements are evaluated to determine how the event and goal combination affects the NPC. Since it is responsible for handling incoming information, the Process Unit needs to be efficient and fast to prevent data loss, leading to the decision to discard information as soon as it is not useful.

6.3.2 Emotion State

The Emotion State is the intermediate unit between event processing and reactive behaviours. It is also where Plutchik's psycho-evolutionary synthesis is implemented. Its main purpose is to store and maintain current NPC emotion values, but it is also the location of the combination rules used to define complex emotions. It has three components: Emotion Store (ES-1), Emotion Decay (ES-2), and Emotion Combinator (ES-3).

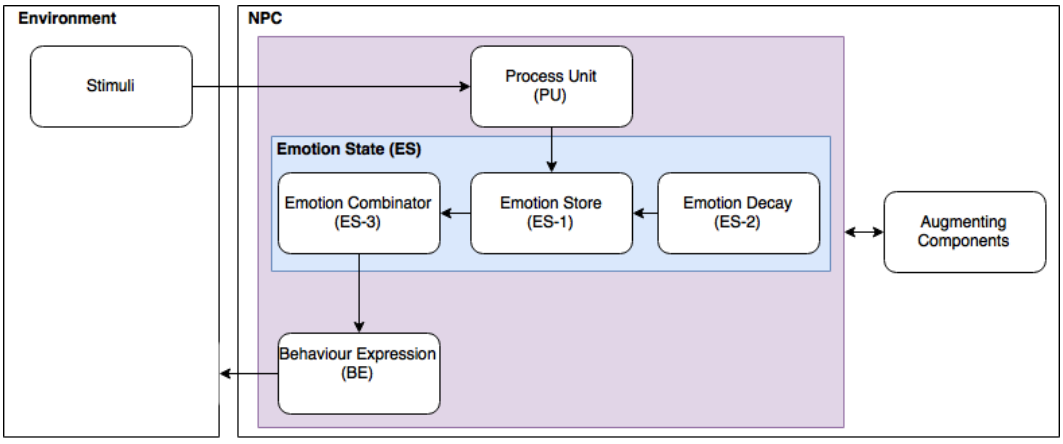


Figure 6.3: Emotion State Components

ES-1. **Emotion Store** (Functional Requirement F–6, F–7, F–8)
Emotion values produced by the Process Unit (PU) first enter the Emotion State via the Emotion Store, a database for storing basic emotion values as defined by Plutchik's psycho-evolutionary synthesis. Each of the eight basic emotions have a value between 0% and 100% intensity, where 0 means that the NPC does not feel that emotion at all and 100 means that the NPC strongly feels that emotion. The basic emotions are arranged in opposing pairs (Table 6.1), which means that each time an emotional value changes, its pair must also change by an inversely proportional value. For example, if "Joy" increases by 15%, then Sadness must decrease by 15%. This allows for the use of a simple mathematical model that is easy to maintain and adheres to the psychological theory that this unit is based on.

Joy	$\longleftrightarrow$	Sadness
Trust	$\longleftrightarrow$	Disgust
Fear	$\longleftrightarrow$	Anger
Anticipation	$\longleftrightarrow$	Surprise

Table 6.1: Plutchik's Opponent-Pairs of Emotion

ES-2. **Emotion Decay** (Functional Requirement F–9)
Human emotions tend to return to an equilibrium state in the absence of additional emotion-inducing stimuli, implying that there is an emotion decay that occurs over time. To simulate this, a function that changes the values in the Emotion Store (ES-1) is triggered periodically until a pre-determined equilibrium state is reached. The form of the function and length of the decay period can vary between NPC specifications. In

addition to this natural decay of emotions, there might be other situations which cause unusual emotional behaviour. For example, joy immediately follows extreme terror in the Opponent-Process theory (Solomon, 1980). The decay functions and equilibrium state values are separated from the Emotion Store module because they are not directly related to Plutchik's basic emotions. It also keeps user-defined rules separate from key Emotion State functions that should generally not be altered.

ES-3. **Emotion Combinator** (Functional Requirement F–10, F–11)
While basic emotions might be enough to specify NPC behaviour in most cases, there will be times when game designers need a behaviour to occur when multiple emotions are at a specific level or at different emotion intensities. Psycho-evolutionary synthesis allows for the combination and intensity of different emotions to result in complex emotions, such as awe (fear and surprise) and rage (high intensity anger). The Emotion Combinator module allows for the specification of these complex emotions such that they arrive as a single emotion type in the Behaviour Expression component as opposed to a collection of otherwise individual emotions.

The Emotion State is the main component where Plutchik's psycho-evolutionary synthesis is implemented. Defining a small set of basic emotions might be enough for most games, and the arrangement into opposing emotional pairs makes the rebalancing of stored emotion values simple and efficient. This design is encapsulated in the Emotion Store (ES-1) component. Another aspect of psycho-evolutionary synthesis, encoded in the Emotion Combinator (ES-3), is the definition of complex emotions as combinations of basic emotions and emotion intensities. These are defined when basic emotions do not produce the desired behavioural complexity. Unmentioned in the psycho-evolutionary synthesis, but still observed in real life, is the gradual decay of emotions to an equilibrium state in the absence of emotion-inducing stimuli. This observation is captured in the Emotion Decay module (ES-2), which is used by the Emotion Store during a pre-determined time cycle. This additional module is also useful in cases where more complex decay functions are required, such as those observed in the Opponent-Process theory. When the Emotion State has finished its computations, it produces a set of emotions and intensities that can be extracted by the Behaviour Expression module to determine how the NPC should react.

6.3.3 Behaviour Expression

The final unit in the GLaDOS system, Behaviour Expression, addresses the second and third components of Lazarus's cognitive appraisal. These sections

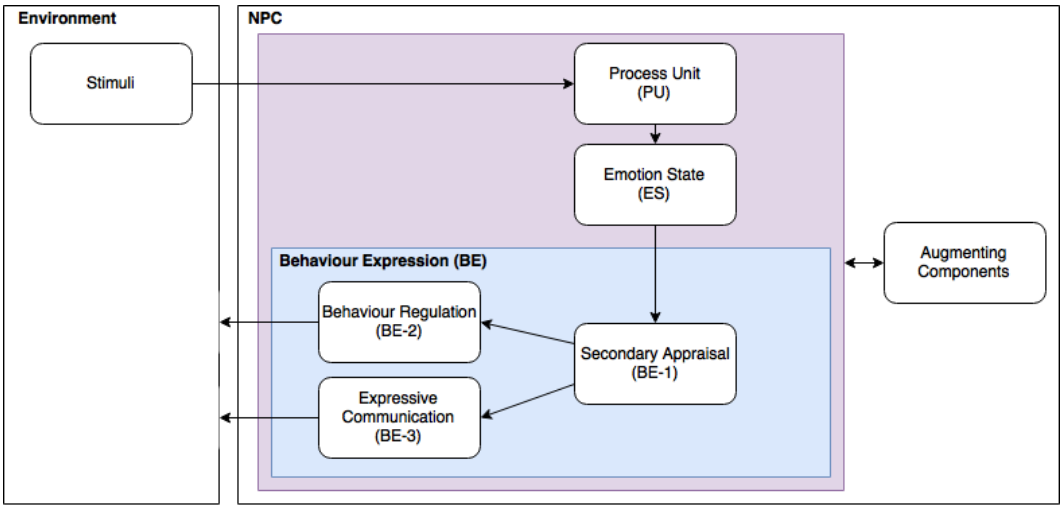


Figure 6.4: Behaviour Expression Components

of the theory are responsible for assigning responsibility or blame for an event and deciding how to react. This component has three modules: Secondary Appraisal (BE-1), Behaviour Regulation (BE-2), and Expressive Communication (BE-3).

BE-1. **Secondary Appraisal** (Functional Requirement F–12)
After emotion values have been updated and combined into complex emotions, all emotions and their intensities are examined by the Secondary Appraisal module. This module checks each emotion/intensity set to determine if it could trigger a change in NPC behaviour or animation according to encoded threshold rules. After examining each set, the largest intensity and associated emotion is selected and passed onto the Behaviour Regulation (BE-2) and Expressive Communication (BE-3) modules. If some emotions defined in the Emotion Combinator (ES-3) require blame assignment, it is determined in the Secondary Appraisal module. This is used to identify a specific target that a behaviour or animation is directed. For example, if Awe is defined as [Target $\rightarrow$ ([Fear $>$ 75] + [Surprise $>$ 75])], then the Secondary Appraisal module directs the response towards "Target".

BE-2. **Behaviour Regulation** (Functional Requirement F–13)
The emotion/intensity set selected by the Secondary Appraisal module (BE-1) is used by the Behaviour Regulation module to select NPC behaviours that require scripted rules, such as moving to a destination or switching to another behaviour state. This is the module where traditional behaviour scripts can be used, allowing for a library of available

scripts to enable an NPC to select alternate behaviours for different situations. This limits the additional overhead traditionally required to handle multiple possible NPC behaviours where a game designer would need to manually assign each script to a selection rule.

BE-3. **Expressive Communication** (Functional Requirement F–14)
Working in parallel with Behaviour Regulation (BE-2), the Expressive Communication module is responsible for selecting and playing animations to match the NPC's current emotional state. Like Behaviour Regulation, the selection space for this module is passed in from the Secondary Appraisal module. The decision to separate scripted behaviours from animations, aside from encapsulating different sets of functionality, addresses the observation that sometimes an emotional state can affect your internal thoughts but does cause behavioural changes. This also allows game designers to specify situations with conflicting behaviours and animations, such as smiling while being attacked.

Behaviour Expression implements the remaining pieces required to create a basic model of Lazarus's cognitive appraisal theory. While the Process Unit extracts emotion values from environmental events, Behaviour Expression determines what can be done about the new emotion state and selects the best actions and emotional displays to play in response. The creation of the selection space is handled by the Secondary Appraisal module (BE-1), which takes in the current emotion state, both emotions and their intensities, and selects new NPC behaviour or animations. When the selection is made, it is passed to the Behaviour Regulation (BE-2) and Expressive Communication (BE-3) modules where the NPC's final reaction, a combination of behaviours and animations, is chosen and manifested in the game environment.

6.3.4 Augmenting Components

The three components of the basic GLaDOS system – Process Unit, Emotion State, and Behaviour Expression – are likely enough for most game designs. However, there are still several other factors that have been identified by psychologists that influence emotions, both in their creation and expression. These factors can be considered additional system components when more complex NPC behaviour is required, or if a more accurate model of emotion is desired.

- **Personality**
  When asked how the same event can cause differing reactions, many people identify personality as the main contributor. Defining personalities for NPCs could cause immediate and noticeable differences in their

behaviour, creating the illusion that a game world's population is diverse and unique. Personality can be implicitly defined in the GLaDOS system via the Behavioural Expression component. In the Secondary Appraisal module (BE-1), a game designer can specify different behaviour thresholds for each emotion and scripted behaviours and animations can be included in the Behaviour Regulation (BE-2) and Expressive Communication (BE-3) modules that are unique to an NPC. However, this methodology requires additional implementation time due to the variety of scripts required. One method of explicitly introducing personality into emotion generation is by creating processing modifiers for the Primary Appraisal module (PU-4) in the Process Unit. If used well, these modifiers can change how events are interpreted, leading to differing emotion values for different NPCs, which in turn could lead to varying behaviours.

- **Mood**
  Mood is another factor that many people identify when asked what influences how events are interpreted and responded to. It could be described as a persistent emotion state that exaggerates some emotions over others. A mood could be simulated by maintaining a separate set of emotion values that stores the cumulative average of past emotion states. This set of values can then be used to influence new emotion values going into the Emotion Store (ES-1) by altering their original value. A mood state could also directly influence the Secondary Appraisal module (BE-1) by altering what emotion/intensity sets to pass along. The result could cause a variation in NPC behaviour based on their short term experiences.

- **Beliefs and Values**
  How people interpret and react to different situations is a culmination of their knowledge and experiences. Beliefs and values are part of the knowledge used to formulate emotional responses. This knowledge is formed over time and can be influenced by social interactions. Stereotypes, where someone expects certain things from another based on observable attributes such as nationality, are an example of a commonly held beliefs. The interesting thing about beliefs and values is that they do not need to be true, which can lead to any number of irrational behaviours. As with personality, beliefs and values can be implicitly included in the GLaDOS architecture via the Behaviour Expression module (BE) by specifying what types of behaviours and animations are available based on the result of a responsibility assignment, and explicitly as modifiers in the Primary Appraisal module (PU-4) in the Process Unit. In cases where beliefs and values are implemented as part of Behaviour Expression, it might be necessary to return the results back to Primary

Appraisal in order to simulate a changing emotional response as more knowledge is assessed.

- **Memories and Images**
  An observable phenomenon is how memories and images affect emotions by reminding an individual of an event, which can trigger an equal or milder emotional change in response. This phenomenon can be integrated into the GLaDOS system by introducing a database for "memories" – past events that caused drastic changes to the NPC's Emotion Store (ES-1). Reasoning mechanisms need to be designed that are able to recognize similarities between a current event and events stored in this database so that it alters the values produced by Primary Appraisal (PU-4).

- **Social Interactions**
  Social interactions are specialized environmental events that require NPCs to comprehend both verbal and non-verbal social signals, formulate a representation of the other party's internal emotion state, and respond in turn. The response does not have to make sense from a social standpoint – it needs to make sense within the context of the situation and the NPC's personality. This extension to the GLaDOS system requires a series of new modules in the Process Unit, arranged in parallel to pre-existing modules. The purpose of a parallel processing branch is to extract and analyse event components that are specific to social interactions, such as language processing and cultural norms, separately from general event information.

- **Social Learning**
  If methods for processing Social Interactions are added to the GLaDOS system, a social learning aspect can also be added. In the absence of others, people might develop habits and routines that might not be considered acceptable within the greater social realm. This leads to a number of changes that include a reduced likelihood of performing undesirable behaviour or stopping it altogether. This can be simulated by implementing a learning algorithm in the Secondary Appraisal module (BE-1), which changes the selection likelihood of certain behaviours and animations.

- **Personality Development**
  It is known that personality can change as a result of experiences, society, and other social factors. In the GLaDOS system, personality can be expressed implicitly through the selection bias of the Secondary Appraisal

module (BE-1) and explicitly by modifying the Primary Appraisal module (PU-4). This implies that it is possible to simulate a changing personality by modifying one of these aspects. However, the method chosen to create changes in personality should be slow, since many personality changes happen over time as experiences accumulate.

- **Neurobiological Systems**
  There are many emotional and behavioural traits that cannot easily be controlled because they are heavily influenced by neurobiological systems. Within the context of a computerized system, neurobiological systems would result in irregular computations and behavioural selections that would not otherwise be made. For example, hormones can be implemented in a module that alter values in the Emotion Store (ES-1) at random time steps. Other brain chemical imbalances can also be simulated in various ways, such as preventing a change in behaviour, causing an exaggerated appraisal of an event, or not accepting new events from the environment. How they are implemented will depend on the biological system itself.

While a basic GLaDOS system might be sufficient for most NPC specifications, there are a number of extensions that can be made to reduce its static nature, potentially resulting in a more interesting experience for observant players. The most likely change that game designers will make is the addition of explicit personality modifiers, which can complement the implicit expression of personality in Behaviour Expression. Another likely augmentation is mood, a culmination of past emotion states that influence how current events are processed. Extensions that require additional memory and management, beliefs, values, memories, and images, can be used to create biased NPCs that make decisions based on past experiences. On a social level, effects from social interactions and learning can also be implemented as a parallel processing unit and learning algorithm, respectively. An emergent personality can also be implemented with a learning algorithm on top of any implementation of personality. These additions can lead to the development of an NPC community over a game's lifetime. The final suggested extension, neurobiological systems, can be used to express otherwise inexpressible behaviours, such as hormonal imbalances, exaggerated appraisals, or ignoring environmental events.

6.4 Conclusion

The main purpose of the GLaDOS design is to enable game designers to create believable and interesting NPC behaviours in CRPGs, with the goal of enhancing player engagement and entertainment. The CRPG genre was chosen for

the initial design because NPC behaviours are highly visible and can impact how the player plays. The psychological theories of cognitive appraisal and psycho-evolutionary synthesis were chosen to see if a single generalized system based on theories is enough, or if a new design is required for each game. Even though the player is the main stakeholder, many of the identified use cases and functional requirements focus on the customization of the GLaDOS system because game designers must be able to customize it to suit their own specifications. However, the player is the main motivator behind several of the non-functional requirements, such as system performance.

The requirements lead to an architecture design with three distinct components (Figure 6.5). The first component, the Process Unit, is tasked with gathering event information, then filtering and analysing it with regards to a set of formalized NPC goals. The second component, Emotion State, tracks the current collection of event processing results as a set of emotion values. These values return to a pre-defined equilibrium state over time via decay functions and can be combined to form complex emotions with user-defined rules. These values are used to influence how behaviours are chosen in the final component, Behaviour Expression. This module is responsible for selecting which subset of behaviours and animations can be used based on emotion categories and trigger thresholds. Behaviours and animations are selected separately to support both modularity and the ability to create otherwise incompatible sets, such as smiling while in pain. Both the Process Unit and Behaviour Expression components implement Lazarus's cognitive appraisal theory while the Emotion State implements the emotion taxonomy presented in psycho-evolutionary synthesis. The selection of two seemingly compatible theories was made because it allows for the most defined aspect of each to be used, instead of having one well developed component in exchange for some underdeveloped ones. There are also several possible psychology-based modifications and extensions that can be made to make NPCs more believable, but they might provide more functionality than is required for most designs. In order to determine if the proposed GLaDOS system design truly meets the goal of increased player engagement and entertainment while being easy for developers to modify and extend, the system needs to be built and tested.

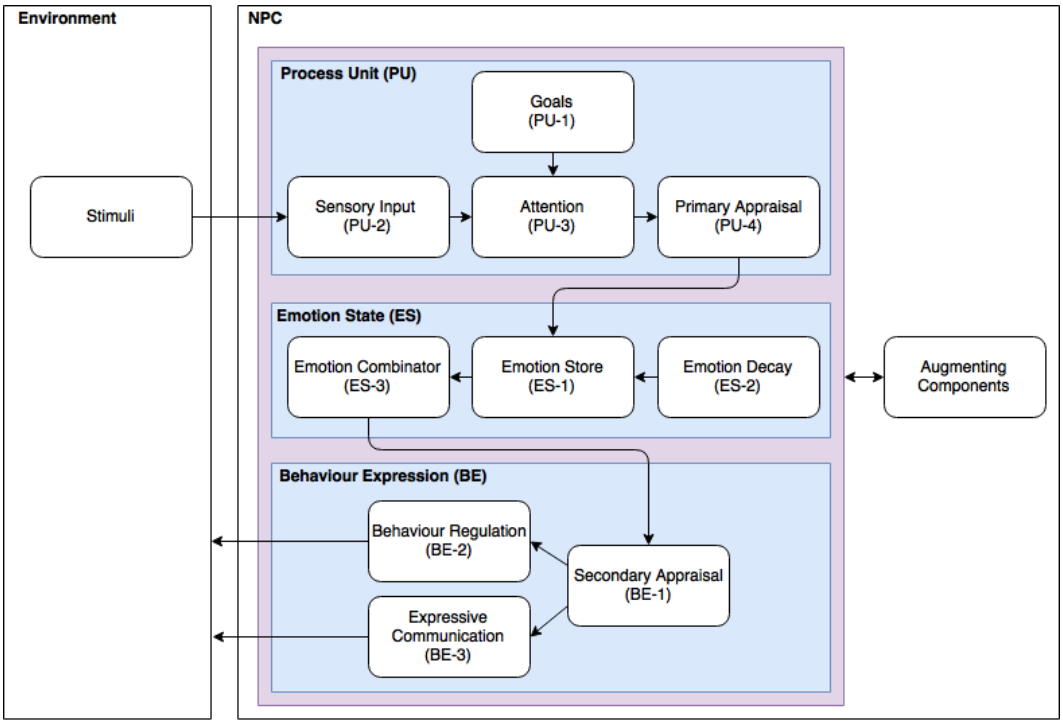


Figure 6.5: Complete GLaDOS Architecture

Chapter 7

Implementing the GLaDOS System

Since the GLaDOS system is designed such that it can be appended to a pre-existing game, this implementation route was chosen because it would eliminate additional content creation tasks required during the game development cycle. The chosen host game, Bethesda's *The Elder Scrolls V: Skyrim*, is known for its support of user-created content[1] and the community that has emerged around this task. It is important to note that some of the required functionality is not included in the core Creation Kit, so a community-created Skyrim Script Extender (SKSE)[2] must also be installed in order for this implementation to function correctly. Although selected for practicality, *The Elder Scrolls V: Skyrim* is a sandbox CRPG. Due to its nature, the scope of sandbox CRPGs is the largest of the three types and allows the GLaDOS system to be implemented in an environment which would typically require the most work to define the design space for. Therefore, this implementation of the GLaDOS system can also be used as a feasibility benchmark for any CRPG.

Translating the components defined in the GLaDOS architecture was relatively simple. Most components could be represented with a single script object, with the exception of the Sensory Input component of the Process Unit (Section 7.3.2) and the Behaviour Regulation component of the Behaviour Expression module (Section 7.5.2). Since these components interface directly with the game, it is not surprising that additional mechanisms were needed in order to connect it with the user-generated system. After the core GLaDOS system was implemented, additional scripts and objects were created as needed for tasks such as system initialization and information flow.

[1] Copies of the development platform, Creation Kit, and associated scripting language, Papyrus, are included with PC versions of *The Elder Scrolls V: Skyrim*

[2] Skyrim Script Extender Homepage

Even though the GLaDOS architecture was designed around specific psychological theories, unspecified architecture details are not which leaves their definition up to implementation requirements and constraints. In some cases, implementation details have been loosely based on additional psychological theories to produce a functional system. Specific decisions made during the implementation process cannot be fully supported by formal models or research, and can be subjected to debate. Despite the limitations of the design specifications, the resulting GLaDOS implementation was created to test if the proposed design could enhance a player's experience, improving the chances that they will replay an otherwise "completed" game. This specific implementation is one of many potential ways to approach the GLaDOS design, and future users are encouraged to alter their implementation strategy to suit their needs.

7.1 Connecting the GLaDOS System to *The Elder Scrolls V: Skyrim*

Implementing the GLaDOS system in *The Elder Scrolls V: Skyrim* required the use of some unusual practices. A Quest object[3], which starts upon game initialization and cannot be completed by the player, is required to be able to access the GLaDOS system for initialization and run routine procedures. The inability to complete this Quest is necessary because associated objects are removed upon its completion. Unless the player wishes to stop the GLaDOS system, its associated objects should not be removed during execution. For simplicity, the Quest object is also named GLaDOS. This Quest object contains a series of ReferenceAlias objects[4] that are assigned to specific NPCs when the quest starts[5]. The ReferenceAlias objects are containers for the various modules that make up the GLaDOS system, which have been configured for each unique NPC. Configuration parameters include NPC-specific values required for the `GPrimaryAppraisal` module (Section 7.3.4) and partially stored in the `GAttention` and `GGoals` modules (Section 7.3.3), emotion decay rate and equilibrium data for the `GEmotionDecay` module (Section 7.4.3), emotion selection thresholds and default selection in the `GSecondaryAppraisal` module (Section 7.5.1), behaviour packages used by the `GBehaviourExpressionObject` module (Section 7.5.2), and animations used by the `GExpressiveCommunication` module (Section 7.5.3). Values were designed to fit within an NPC's known characteristics and personality

[3]Creation Kit Wiki – Quest Object (Editor)

[4]Creation Kit Wiki – ReferenceAlias Object (Quest Object)

[5]A small test class, `GTestFill`, is included to be able to test selection conditions in ReferenceAlias definitions to ensure that the intended NPC is being picked.

as described by the game's documentation[6]. This organization provides two advantages: any change to the underlying system logic is propagated to all ReferenceAlias objects, and any changes made to individual ReferenceAlias objects do not affect other ReferenceAlias objects.

A separate ReferenceAlias object for the player is also required for the information gathering GLaDOS system components (Section 7.3.2) and detection system (Section 7.3.1). This inclusion makes it possible for the GLaDOS system to recognize and process the player's current state to produce behavioural changes and animations.

Despite the odd practices, the implementation of the GLaDOS system closely follows the intended design. The Process Unit (Section 7.3) gathers information from the game environment and converts it into emotional values; the Emotion State (Section 7.4) stores and updates the current system emotion values using new values from the Process Unit and uses decay functions to return values to an equilibrium state; and Behaviour Expression (Section 7.5) determines what actions and animations to execute based on the values extracted from the Emotion State. To ensure that the components work together, a control class was implemented.

7.2 Controlling the GLaDOS System (`GLorthiem`)

The main control class is considered part of the interface between the game environment and the GLaDOS system (Figure 7.1). Nicknamed "Lorthiem", after the NPC used to test the GLaDOS system, it initializes required system components when the NPC is created, notifies the system of scheduled updates, and coordinates inter-module communications.

System initialization includes the extraction and assignment of system module references from the NPC's ReferenceAlias to script variables, improving long-term system performance. This start-up logic is stored in the *Initialization* function rather than in the *OnInit* event[7]directly, allowing setup logic to be performed during non-standard NPCs initialization (Section 7.6). Further initialization tasks are handled in the first *OnUpdate* event[8] including the creation of a `GProfile` object in the `GSensoryInput` module (Section 7.3.2), the assignment of equilibrium emotion values in `GEmotionStore` from values stored in `GEmotionDecay` (Sections 7.4.1 and 7.4.3), and the recording of the NPC's default Energy Level variable for use with AI packages

[6]The Elder Scrolls Wiki and Unofficial Elder Scrolls Pages were used for this purpose.
[7]All objects in the Creation Kit have this event as part of their core logic.
[8]Creation Kit Wiki – OnUpdate() Event (Form, Alias, and ActiveMagicEffect Scripts)

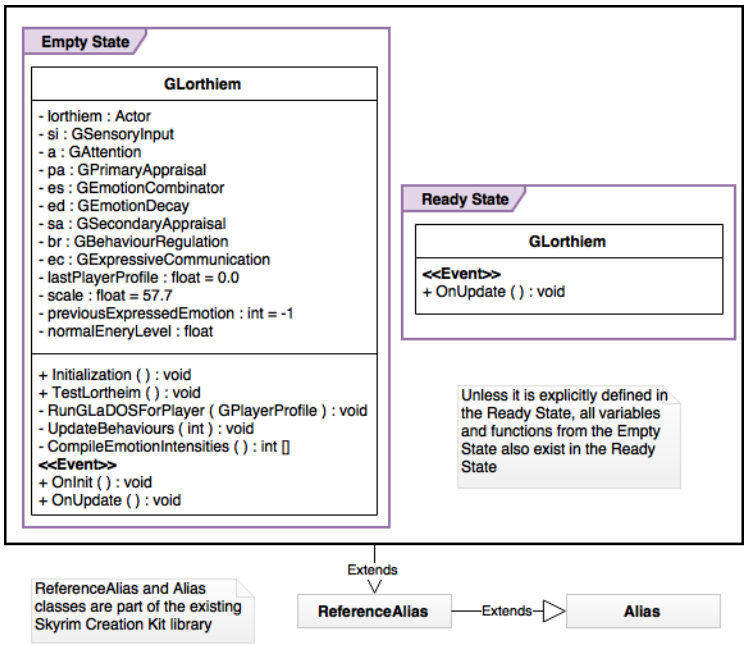


Figure 7.1: Internal Control Class

(Section 7.5.2). This event is added to the processing queue after initial variable assignments are made. These tasks were moved to an *OnUpdate* event to avoid a known bug where some game events are not registered while the *OnInit* event is running.

The *OnUpdate* event is also required for different tasks post-initialization. Therefore, two versions were created via the use of states[9], which enables the use of pseudo-polymorphic function definitions. The version used for initialization defined in the system's initial state, the "Empty State". After completing the first *OnUpdate* event, the system transitions to the user-defined "Ready State", where the second definition of *OnUpdate* is located. The state change does not affect anything else in the `GLorthiem` class because they are not defined in the "Ready State". This is the result of the Creation Kit's state system design.

The "Ready State" *OnUpdate* event is called for the first time approximately one second after the state change and remains in this state for the remainder of the system's running time. The *OnUpdate* event in the "Ready" state is responsible for periodically running the `GEmotionDecay` module according to its specified decay step, and checking for any changes to the expected expressed emotion state via the `GSecondaryAppraisal` module

[9]Creation Kit Wiki – States

(Section 7.5.1). If a change occurs such that a different emotion needs to be expressed, a follow-up call to the *UpdateBehaviours* function is made, which is responsible for coordinating the `GBehaviourRegulation` (Section 7.5.2)and `GExpressiveCommunication` (Section 7.5.3) modules.

The final task of the main control module is the coordination of communication between system modules. This has a twofold design purpose. The first is to remove direct dependencies between classes, allowing future developers to alter modules without making cascading updates to additional modules. The second purpose is to aid in performance by allowing certain classes to run relevance checks before proceeding, such as running the `GAttention` module before solving otherwise irrelevant Threat Control equations in the `GSensoryInput` module.

When the system is triggered, it compares the creation time stored in `GPlayerProfile` (Section 7.3.2)[10] with the recorded creation time in `GLorthiem`. If the profile object's creation time is later than the stored one, the GLaDOS system proceeds with analysis. Limiting profile analysis to one time per update avoids scenarios where an NPC acquires identical player profile objects in a short time span. Analysing each one would result in wasted computational resources and artificially compounded emotion values in `GEmotionStore`. Once the profile object is marked for processing, the NPC is forced to look at the player to signal that the process has begun. The system then extracts and compares the profile stimuli dedication values, a percentage value representing how many times a stimuli appears in a profile, with the NPC-specific threshold values of `GAttention`. Three stimuli are selected[11] and are used to extract three sets of NPC-specific goal information from `GGoals` (Section 7.3.3). This is also the stage where game-specific values are calculated and important player-held item information is gathered. The collection of information, goals, calculations, and items, are passed as a set to `GPrimaryAppraisal` (Section 7.3.4) so that they can be converted into an emotion and accompanying intensity value. Since the equations in `GPrimaryAppraisal` are normalized, the intensity value is translated into the range $[0,100 \in Z]$ after the computations are complete. At this point, the resource-intensive computations have been complete, but the new values still need to be stored and translated into game environment output. Adding the new intensity value to `GEmotionStore` is handled via a selection structure to identify the emotion code and send the intensity value in through the associated channel. Following the update to `GEmotionStore`, the new emotion

[10]For this implementation, NPCs are only able to extract a player profile object. In future implementations, NPC profile objects might also be candidates for analysis.

[11]The player's race is always included in the three stimuli in this implementation because its dedication value is always 1.0, the maximum value.

state is sent to `GSecondaryAppraisal` to determine which, if any, expression threshold the new state crosses. The result of this appraisal is sent to the *UpdateBehaviours* function so that any associated behavioural or animation changes can occur. After the system finishes, the NPC is no longer forced to look at the player to signal its completion.

The inclusion of the `GLorthiem` control class is essential to the smooth operation of the GLaDOS system. It ensures that all required elements are initialized properly and in a manner that is the least intrusive to other game tasks. It controls the timing of changes and updates to NPC behaviours due to the `GEmotionDecay` module. It also moderates the flow of information between GLaDOS components during an encounter. Separating these tasks from the core system modules via `GLorthiem` allows more flexibility in how and when the system operates.

7.3 Process Unit

The Process Unit (Figure 7.2) is where game environment data enters the GLaDOS system. Gathered information is filtered according to pre-designed NPC goals before being synthesized into numerical values that can be encoded as an emotion and intensity value. The synthesized values are then used by the system to determine the new NPC emotion state (Section 7.4). Implementing this process in *The Elder Scrolls V: Skyrim* required the creation of four distinct information processing tasks: gathering (7.3.1), generation (7.3.2), conditioning (7.3.3), and analysis (7.3.4).

7.3.1 Information Gathering (The Detection System)

In order to use the GLaDOS system, NPCs need to be able to detect targets and begin processing their associated stimuli. A detection system already exists in *The Elder Scrolls V: Skyrim* that is based on Line Of Sight (LOS) events which are triggered whenever the observing agent can draw an unobstructed line to a target. However, these events are known to be computationally expensive and are throttled by the game engine[12]. This causes increasingly large delays between event registrations, which might result in a delayed start of an NPC's system. This could cause delayed reactions, sometimes even after the stimulus has been removed, or no reaction at all. This approach is not ideal for resource usage reasons as well. Since the system will only be triggered when the player is within a set distance of an NPC, it is not resource-efficient to register all NPCs for LOS events at all times. Even if an NPC is within detection range of the player, the LOS events will constantly check if the player

[12]Creation Kit Wiki – RegisterForLOS (Form Script Function)

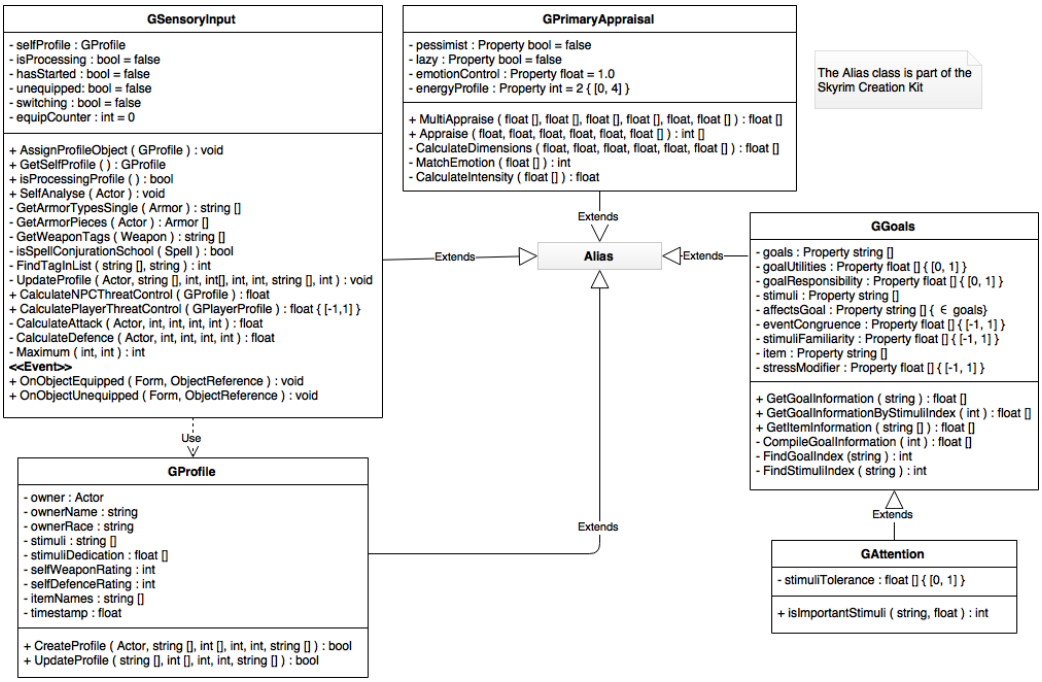


Figure 7.2: Process Unit Classes

can be seen, even if neither is moving, resulting in further wasted resources. To be feasible for this type of use, a detection system should be able to do a quick, general check to see if the player is within range before deciding if a LOS can be established. A community-generated solution, originally created to dynamically attach scripts to NPCs[13], has been modified for this purpose because it uses minimal resources while still using the available LOS capabilities. The detection system is built using objects created from the Spell[14], MagicEffect[15], and ActiveMagicEffect[16] classes that are included in the Creation Kit (Figure 7.3), and is comprised of three main parts: a cloak that defines the detection radius, a procedure that determines if the GLaDOS system should be activated based on LOS, and an interface for connecting to the player's avatar.

Defining the Detection Radius (`GLaDOS_DetectionSpell` and `GLaDOS_DetectionRange`)

The first set of detection system components defines a field around the player to determine which NPCs are close by. This enables control of the maximum

[13]Creation Kit Wiki – Dynamically Attaching Scripts (Tutorial)
[14]Creation Kit Wiki – Spell (Object)
[15]Creation Kit Wiki – MagicEffect (Object)
[16]Creation Kit Wiki – ActiveMagicEffect (Papyrus Script)

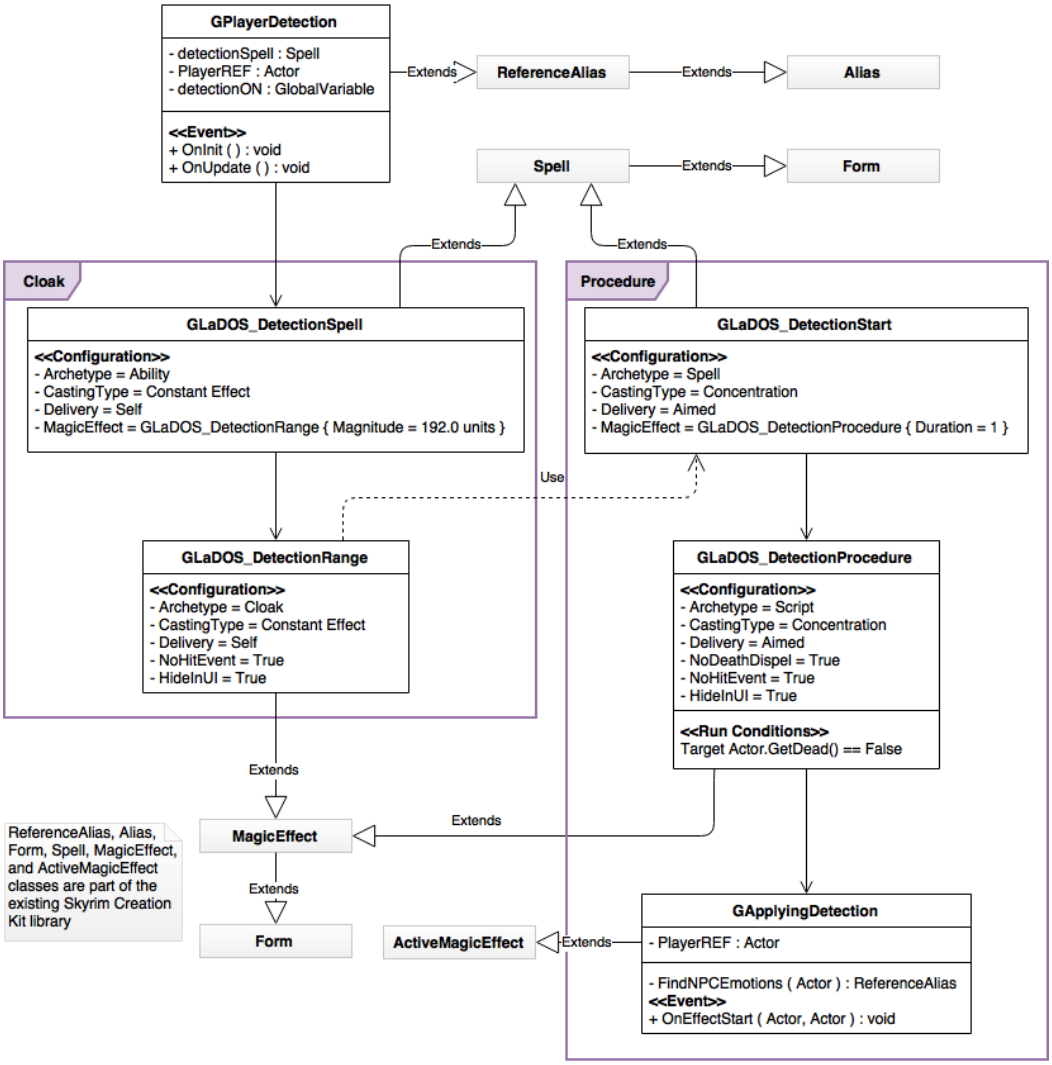


Figure 7.3: Detection System Classes

distance that an NPC can be from the player and still reasonably "see" them. This feature is not possible with standard LOS events, which could result in an NPC detecting and analysing the player from an unrealistically large distance in environments such as fields. The detection field is defined by four attributes: activation method, location in the game environment, shape, and size.

The activation mechanism and location of the detection field within the game environment is defined by attributes in `GLaDOS_DetectionSpell`, an object implementation of the built-in Spell class. Ideally, the detection field should be activated passively and centre on the player. This is handled by assignments to the *Archetype*, *Casting Type*, and *Delivery* fields. The value assignments for both *Archetype* and *Casting Type* are built-in types, and were

selected because they cause the GLaDOS_DetectionSpell to run continuously without player intervention. The target of this object, defined with the *Delivery* assignment, is the agent that holds this object. Since this object is assigned to the player, this translates to the detection field being centred on them. With the detection field activating passively and encompassing the player, the size and shape of the field can be defined.

The detection field's size and shape are defined by creating a MagicEffect object using the built-in class, and associating it with the GLaDOS_DetectionSpell object. The MagicEffect object, GLaDOS_DetectionRange, contains the field's shape and additional object flags that control how it interacts with other in-game elements, such as not appearing in the user interface (UI) and appearing as a non-hostile action. The optimal shape for a detection field is a sphere since its sides are equidistant from the sphere's centre in all directions. This shape can be achieved with the Cloak archetype that can be assigned to a MagicEffect object. Finally, the size of the detection field is defined when the GLaDOS_DetectionRange MagicEffect object is associated with the GLaDOS_DetectionSpell Spell object. As part of the association process between the MagicEffect and Spell objects, the *Magnitude* of a MagicEffect object can be defined to reflect the maximum detection distance. For this implementation, a detection radius of 192 units[17] was used, which is approximately the length of a cell in *The Elder Scrolls V: Skyrim*. Due to its dependence on the GLaDOS_DetectionSpell object, the GLaDOS_DetectionRange object was assigned identical *Casting Type* and *Delivery* values.

At this point the location, size, and shape of the detection system have been defined. The GLaDOS_DetectionSpell Spell object data is used to centre the detection system on the agent that is holding it. In this implementation, only the player is given this object so the detection field is centred on them. The shape of the detection system, a sphere, is determined by the GLaDOS_DetectionRange MagicEffect object. The association of this MagicEffect object with the GLaDOS_DetectionSpell object allows the size of the detection field to be determined. This alone is not enough to implement a functioning detection system because the logic that determines if an NPC has a LOS to the player must still be defined.

Determining LOS (GLaDOS_DetectionStart, GLaDOS_DetectionProcedure, and GApplyingDetection)

Once an NPC has made contact with the detection field, the system must decide if the NPC can "see" the player and initiate the GLaDOS system. Unlike the detection field, which makes a general check to see if any NPC

[17] Approximately 192ft (1 unit $\approx$ 0.5625 inches).

is close to the player, determining if a specific NPC should begin processing is a targeted process. This means that a targeting system must be included as part of the LOS check so that appropriate pre-processing checks can be made. For this section of the detection procedure, three attributes must be defined: a target selection mechanism, LOS checks, and a method for calling the GLaDOS system.

Target selection is handled by associating a targeted Spell object, `GLaDOS_DetectionStart`, with the `GLaDOS_DetectionRange` MagicEffect object from the detection field, causing it to activate when an NPC connects with the detection field. Its *Archetype* is set to Spell because it does not require the specialized functionality offered by the other options, and could potentially be hindered by some of their usage constraints. In addition to being useful for targeted NPC selection, the *Casting Type* and *Delivery* fields must be Concentration and Aimed[18] respectively, in order to be eligible for association with `GLaDOS_DetectionRange`. Now that the detection system is able to target NPCs that come into contact with the detection field, the LOS checks must be established.

The required LOS tests can be performed via script functions, but it is not possible to directly associate a script with a Spell object. Therefore, an intermediary MagicEffect object, which can be associated with both Spell objects and scripts, was created. The `GLaDOS_DetectionProcedure` MagicEffect object was assigned its *Casting Type* and *Delivery* values to meet the Spell object association requirements, and its *Archetype* for script association. As part of the Spell association definition, the `GLaDOS_DetectionProcedure` MagicEffect was assigned a *Duration* of one second to ensure that it will be registered by the affected NPC. The Script *Archetype* also ensures that the functionality defined in the accompanying script object is only invoked when an NPC is affected by this MagicEffect object, as opposed to effects that are inherent to the other *Archetype* classifications. Similar to the `GLaDOS_DetectionRange` object, flags were set to ensure that this object does not appear in the UI, nor registered as a hostile act, which could cause some NPCs to attack upon contact with the MagicEffect.

One test that can be performed directly from the `GLaDOS_DetectionProcedure` object is to determine if an NPC is "dead". It is possible in *The Elder Scrolls V: Skyrim* to affect dead NPCs, which makes this test an important step in determining if the GLaDOS system should be run. If an NPC is not flagged as dead, then further script-based tests are performed to determine if a LOS can be established between the NPC and the player. In the same vein, an additional flag was set on the `GLaDOS_DetectionProcedure` object that allows it to be run on "dead" NPCs. While this is counter-intuitive, it

[18]There is no projectile object assigned to this Spell object, which generates a warning in the Creation Kit.

allows the required logic to be managed manually, preventing known Creation Kit errors where the "dead" flag is improperly read.

Once a targeted NPC has passed the initial checkpoint, more specific tests are performed before an NPC runs the GLaDOS system. These additional checks are performed via the `GApplyingDetection` script. This script is designed such that it runs when its parent object, the `GLaDOS_DetectionProcedure` MagicEffect, connects with an NPC. Unlike other objects in the detection system, `GApplyingDetection` should only run after it has been applied to an NPC. Therefore, it is extended from the ActiveMagicEffect[19] class, which is used to define effects that have been applied to an Actor, as opposed to the previously referenced MagicEffect class, which is used to define effects before they have been triggered. The first test that this script performs is the *IsDetectedBy* function[20], which determines if the player has been detected by the targeted NPC[21]. Running this function first ensures that the target NPC's detection system is running correctly before an explicit LOS check is run via the *HasLOS* function[22]. If a LOS is established between the targeted NPC and the player, the NPC extracts the player's GLaDOS profile (Section 7.3.2) from their ReferenceAlias for processing.

This completes the GLaDOS detection process. When an NPC touches the detection field, they are targeted by the associated `GLaDOS_DetectionStart` Spell object. Upon contact with the NPC, the Spell object initiates a series of tests to determine if it is appropriate to run that NPC's GLaDOS procedure. These tests are contained within the `GLaDOS_DetectionProcedure` MagicEffect object and the `GApplyingDetection` script. The first test, performed by `GLaDOS_DetectionProcedure`, is to determine if the NPC is alive. If they are not, they should no longer be interacting with the game environment and no further processing is done. Subsequent checks, one to determine if the player has been detected at all, and another to determine if a LOS can be established between the player and the targeted NPC, are performed by `GApplyingDetection`. If an NPC passes all three tests, the player's GLaDOS profile is extracted and the NPC begins the GLaDOS process. With a functional detection system in place, it still needs to be attached to the player so that it can be run.

[19]Creation Kit Wiki – ActiveMagicEffect (Script)
[20]Creation Kit Wiki – IsDetectedBy (Actor Script Function)
[21]Detection is determined by sound, skill, and LOS.
[22]Creation Kit Wiki – HasLOS (Actor Script Function)

Interfacing with the Player Avatar (`GPlayerDetection`)

Even though the detection system has been defined, it cannot run without a subject. To connect the detection system with the player, the `GPlayerDetection` ReferenceAlias was defined and attached to the player's existing Quest Alias object. This script is used to periodically toggle the detection system on and off for resource management purposes and to ensure that NPCs are checked for LOS to the player on a regular basis. This ensures that an NPC is not exempt from further detection checks if they have already been touched by the detection field. If the field was always on, an NPC would only be checked once – the first time it is touched by the cloak – which is not ideal for a game where a player can encounter the same NPC multiple times. The detection system is regulated by two game-dependent cycle periods, both set in seconds: the time in which the detection system is on, and the time in which it is off. The on/off durations are defined separately to allow for better resource management control. A master switch was also implemented as part of the `GPlayerDetection` script which can be used to turn the detection system, and by extension the processing component of the GLaDOS system, off. This switch can be used to reset the system if unexpected behaviours are encountered as a result of running the detection system continuously.

The Detection System as a Gateway to the GLaDOS System

The detection system design is composed of a cloaking spell, which defines an effect radius around the caster, the detection procedure, which performs a two-tiered check to determine if the NPC has a LOS to the caster, and an interface that connects the detection system to the player. Although this approach is more complicated than simply registering for LOS events, the detection system is more resource-friendly because it enables designers to control the detection radius and the execution frequency. This is an advantage over only controlling how often the procedure is executed, especially when it is known that checking for LOS is resource-intensive and might result in unexpected behaviours. The detection system also acts as the gateway into the GLaDOS system, determining when and on which NPCs it is to run. If an NPC has been identified, the main GLaDOS system is called and proceeds to the information gathering stage.

7.3.2 Information Generation (`GSensoryInput`, `GProfile`, `GPlayer`, and `GPlayerProfile`)

As part of the information gathering procedure, two modules were created: `GSensoryInput` and `GProfile`. The `GSensoryInput` module is responsible for generating a `GProfile` object for the NPC it is attached to and

extracting a `GProfile` object from other NPCs to be analysed. Generating the `GProfile` object required the extraction of GLaDOS system-specific tags and game-specific battle rating information from armor, clothes, weapons, and spells that the NPC had equipped, in addition to recording their name, race, and the profile's creation time. To help manage item information, dedications represented as percentages (0,1], denoting how much of the NPC's equipment share that tag, is recorded instead of how many of each tag is counted[23]. There are eight equipment slots[24] and NPCs are always assigned a race, meaning that it is possible for an NPC to have at least nine unique tags for this implementation of the GLaDOS system.

Once the information generation task is complete, the only data required to use the system is two `GProfile` objects – one from the observing NPC and one from the observed NPC. In the case where an NPC changed their equipment, either by removing an item or switching one item for another, a new `GProfile` object is created to replace the out-dated one. This has a low impact on system performance because NPCs rarely change their equipped items. There are, however, some noticeable effects when the GLaDOS system is first started because all affected NPCs perform their initial profile generation during system initialization.

Two objects, `GPlayer` and `GPlayerProfile` were created specifically for the player, but perform the same tasks as `GSensoryInput` and `GProfile`, respectively (Figure 7.4). This was done to allow future development to be more flexible, where the player might be given additional functions that the developer does not want to give to NPCs. Players tend to change their items frequently, so information generation might be called several times for them in a single session. However, since the player is often the only one changing their equipment, the performance impact from the player is minimal.

7.3.3 Information Conditioning (`GGoals` and `GAttention`)

When an observed NPC's `GProfile` object is extracted by the observing NPC's `GSensoryInput` module, stimuli dedication values are extracted and passed to the `GAttention` module, an extension of the `GGoals` module, to determine which, if any, of the identified stimuli concern the observing NPC. Concern is determined by comparing GLaDOS system tags with the NPC's list of goals, which are designed to match their personality traits and history. This includes the observed NPC's GLaDOS system tag list and assigned in-game

[23]Tags extracted from full-body armor and two-handed weapons were counted twice. Body armor covers more of the NPC's body than any other wearable item and two-handed weapons are counted as if they were two separate weapons.

[24]Head, Body, Hands, Feet, Neck, Finger, Left Hand, and Right Hand

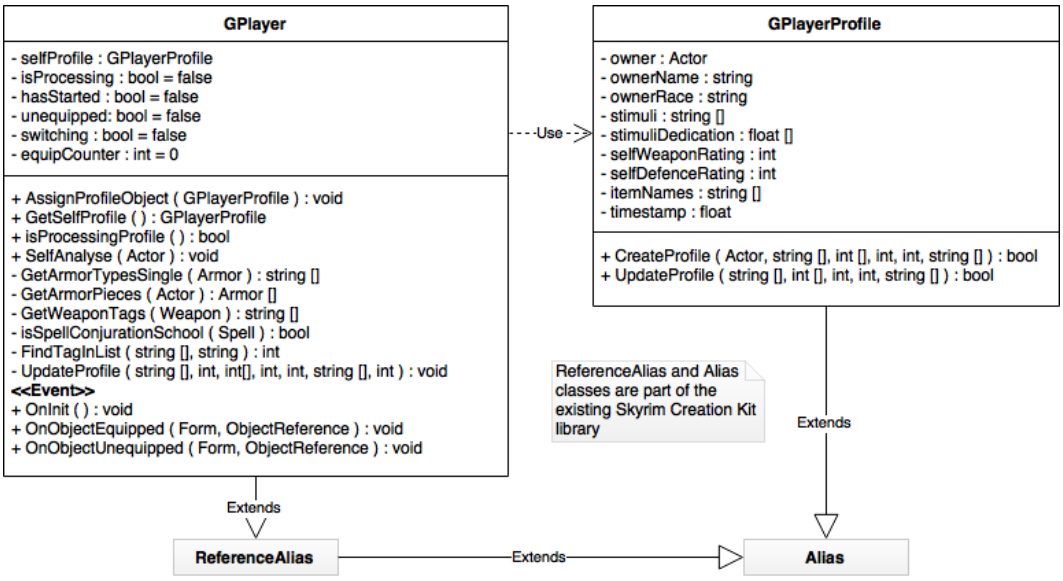


Figure 7.4: Player Classes in the Process Unit

race. Important stimuli are identified by comparing the dedication values[25] from the profile object with the NPC-specific tolerance values, [0,1], in the `GAttention` module. A maximum of three stimuli that meet the tolerance threshold are selected and the associated information is retrieved from the `GGoals` parent module.

Three types of information are retrieved that were designed for each NPC's known personality traits and history. The first concerns the stimuli itself, including the NPC's familiarity [-1,1] with it, which of the NPC's goals is affected by this stimuli, and its congruence [-1,1] with the goal's success. The second type of information is about the affected goal: its utility [0,1] and how responsible [0,1] the NPC feels regarding its success. The third type of information, item stress values, each being a value in [-1,1], is not directly related to specific stimuli or goals. This information was included to account for specific items to be used in appraisal calculation, regardless of their GLaDOS system tags. This enables designers to identify specific items that NPCs are known to be interested in, favourable or not, and ensure that they are accounted for. For example, if an NPC has a legendary item equipped and they are observed by another NPC who studies that item for a living, the item itself should have an impact on the observing NPC's calculations regardless of the system tag associated with it. In this implementation, NPCs can have up to five items that they can uniquely identify from another NPC's `GProfile` object. The collected information from the `GGoals` module includes up to three sets of

[25] An NPC's race is given a dedication value of 1 and is always analysed first. This ensures that one of the selected stimuli for further analysis is the NPC's race.

stimuli and goal data and one set of specific item data that is passed to the `GPrimaryAppraisal` module for analysis.

7.3.4 Information Analysis (`GPrimaryAppraisal`)

The final module in the Process Unit, `GPrimaryAppraisal`, synthesises the gathered data into an emotion and intensity set. To avoid the need to explicitly associate individual game elements with emotions, three dimensions were designed that could be represented by calculations which integrate both native game and GLaDOS-specific values. Even though they were inspired by the Pleasure-Arousal-Dominance (PAD) model of temperaments (Mehrabian, 1996), these dimensions were created because they appeared to be easier to understand, formalize, and attribute to different emotions within the context of CRPGs[26]. This decision also removes the implicit expectation to recreate an exact psychological model, which might be detrimental to either the overall enjoyment of the game or system performance. There also does not appear to be any consensus on the formalization of stimuli decomposition into dimensional values, making the implementation of specific psychological dimension models significantly less valuable. The identified dimensions are:

1. *Desirability* $\in$ [-1,1], a measure of favour towards a stimuli in relation to an affected goal held by the NPC. Positive values represent an affinity towards a stimulus whereas negative values represent an aversion. A value of zero means that the NPC has neither an affinity for, nor an aversion to, the stimulus.

2. *Controllability* $\in$ [-1,1], a measure of the perceived control the NPC has over the outcome of an encounter with the stimulus. This value is a combination of the NPC's ability to manage their emotions and stress, in addition to their perception of their ability to control the posed threat via combat. Positive values mean that the NPC feels confident that they can comfortably handle an encounter with a stimulus, and negative values mean that the NPC feels that they could not manage an encounter with the stimulus. A value of zero means that the NPC perceives that they have a comparable amount of control with the stimulus over any encounters that might ensue.

3. *Effort* $\in$ [-1,1], a measure of how much energy the NPC is willing to exert in response to the stimulus. A large positive or negative value does not necessarily mean that the NPC is experiencing a strong positive or negative event. It does, however, represent a desire to change the current situation in response to the triggering stimuli. Positive values mean that

[26]Pleasure $\rightarrow$ Desirability, Arousal $\rightarrow$ Effort, Dominance $\rightarrow$ Controllability

the NPC wants to exert themselves in response to the stimulus whereas negative values mean that an NPC does not want to exert themselves. A value of zero means that the NPC does not feel that they have to exert themselves.

The resulting mathematical signs (+/−) of the calculated dimensional values are uniquely combined to abstractly represent the basic emotions in Plutchik's taxonomy (Table 7.1). Specific combinations were loosely matched with the eight temperaments[27] of the PAD model, similar to the way that emotions from the OCC model were matched with PAD space temperaments in ALMA (Bock, 2009). The exact value of the individual dimension calculations are used to determine the emotion's intensity (Section 7.3.4).

		Dimensions		
	Code	Desirability	Controllability	Effort
Joy	0	+	+	+
Sadness	1	−	−	−
Trust	2	+	+	−
Disgust	3	−	+	−
Anger	4	−	+	+
Fear	5	−	−	+
Anticipation	6	+	−	−
Surprise	7	+	−	+

Table 7.1: Emotion codes and dimension sign assignments

Calculating the Desirability Dimension

The calculation of *Desirability* (Eq 7.3.1) from GAMYGDALA (Popescu *et al.*, 2014) is replicated here because it is a simple but effective way of formalizing how valuable a stimuli is when compared to an NPC's goals. The calculation requires a *Goal Utility* $\in$ [0, 1] representing its importance to the observing NPC and an *Event Congruence* $\in$ [-1, 1] to describe a stimulus's alignment

[27] Exuberant → Joy, Bored → Sadness, Relaxed → Trust, Disdainful → Disgust, Hostile → Anger, Anxious → Fear, Docile → Anticipation (Anticipation was sometimes referred to as Acceptance in the literature), Dependent → Surprise (this was the only pair left after matching the rest of the temperaments to Plutchik emotions, so this final combination of signs can be viewed as a "don't care" condition.)

with affected goal's maintenance or fulfilment. This information is gathered during the information conditioning process (Section 7.3.3). Negative values mean that a stimulus adversely affects a goal, and is therefore undesirable. The range of utility and congruence values enables an NPC to have reactions with varying intensities depending on the importance of the affected goal and the alignment of the observed stimuli.

$$Desirability \in [-1, 1] = GoalUtility \times EventCongruence \tag{7.3.1}$$

Calculating the Controllability Dimension

Controllability, or perceived controllability, is a compound value comprised of the NPC's ability to regulate their emotions, potential threats, and the perceived outcome of the encounter (Equation 7.3.2). For this implementation, the capacity to regulate personal emotions is represented by a constant value, *Emotion Control* $\in (0, \infty)$ that is designed for each individual NPC's personality. Realistically this value is influenced by several factors in the environment which are beyond the scope of the GLaDOS design, including the NPC's personal beliefs and past experience. The value for *Emotion Control* is defined as a `GPrimaryAppraisal` parameter and is assigned a default value of one. When the value for *Emotion Control* is less than one, the NPC has a more stable personality which causes *Controllability* to approach zero. Conversely, if the NPC's *Emotion Control* parameter is greater than one, their personality is less stable and causes their *Controllability* value to approach the equation boundaries. While it is not immediately intuitive, this does follow the expected outcome where values that remain close to zero cause a low intensity reaction and extreme values cause a high intensity reaction. The sign of the *Controllability* calculation is unaffected by *Emotion Control* since it is always a positive value, but it allows the range of *Controllability* to be $[-\infty, \infty]$. To force the resulting values into the expected range of [-1,1], the $\arctan(x)$ function is used to map $[-\infty, \infty]$ to $[-\frac{\pi}{2}, \frac{\pi}{2}]$, which can then be forced into the [-1,1] range by multiplying by $\frac{2}{\pi}$.

$$Controllability \in [-1, 1] = \frac{2}{\pi} \arctan(C)$$

where

$$C = EmotionControl \times \left(\frac{ThreatControl + PredictionBias}{2} \right) \tag{7.3.2}$$

In this implementation, the calculation of the observing NPC's ability to manage a potential threat, *Threat Control* (Equation 7.3.3), is combat-based depends on two components. The first component, *Attack Value*, is a measure of the observing NPC's aggression level and offensive combat power compared to the observed NPC's defensive combat power. The second component, *Defence Value*, measures the observing NPC's confidence level and their defensive combat power compared to the observed NPC's offensive combat power. This equation was designed to only use values that are already present in *The Elder Scrolls V: Skyrim* because there is already sufficient information contained within the game itself to make a reasonable estimate of the observing NPC's capabilities to manage potential combat threats. The parameters of *ThreatControl* were limited to combat scenarios because it is the only type that does not require additional processing overhead. Trivially, *Threat Control* is maximized when ($AttackValue = 1 \wedge DefenceValue = 0$), minimized when ($AttackValue = 0 \wedge DefenceValue = 1$), and zeroed when ($AttackValue = DefenceValue$). The decision to make *Attack Value* inherently positive was made because it can be viewed as the active state, where the NPC can take action and therefore has more control over the situation. This is contrasted with the passive state of defence, where actions are performed on the NPC and therefore have little or no control over the situation.

$$ThreatControl \in [-1, 1] = AttackValue - DefenceValue \tag{7.3.3}$$

The first half of the equation, *Attack Value* (Equation 7.3.4), is composed of three parts. The first part creates a ratio between the observing NPC's weapon power $\in$ [4,32] and the highest weapon power in the encounter wielded by either the observing or observed NPC. Selecting the largest attack power in the encounter scales the resulting ratio to the power of the individual entities as opposed to a global maximum. This makes differences in offensive power more significant in scenarios where neither NPC has a strong weapon when compared to all weapons in the game, but one is comparatively stronger. The ratio is used to scale *Attack Value* to the relative strength of the observing NPC's offensive power, which should have the largest impact on their perception of the observed NPC's capabilities.

$$AttackValue \in [0, 1] = WR \times \big(AttackRating + AttackProspect \big)$$

where

$$WR = \frac{SelfWeaponPower}{\max(SelfWeaponPower, OpponentWeaponPower)} \tag{7.3.4}$$

The second component of the *Attack Value* calculation is *Attack Rating* (Equation 7.3.5), which measures the observed NPC's ability to defend themselves. Similar to the ratio used for weapon power, the observed NPC's defensive power is calculated as a ratio of their base defence power $\in$ [0,567] and the highest defence power in the encounter. This ratio will be one if the observed NPC's defence is higher than the observing NPC. Therefore, the ratio is subtracted from one to reflect the expectation that a higher opponent defence reduces the observing NPC's ability to break them, lowering their overall offensive capabilities. For this design, the *Attack Rating* calculation is given a weight of 0.7 (70%) in the calculation of *Attack Value*. Weights were used in this case because of the use of aggression in the *Attack Prospect* equation, a quality that can still result in a non-zero *Attack Value*, even if the NPC has no offensive capabilities.

$$AttackRating \in [0, 0.7] = 0.7 \times OD$$

where

$$OD = 1 - \frac{OpponentDefencePower}{\max(SelfDefencePower, OpponentDefencePower)} \tag{7.3.5}$$

Attack Prospect (Equation 7.3.6) is the final component of *Attack Value*, which uses the observing NPC's aggression level[28] which is already present in their original design for *The Elder Scrolls V: Skyrim*. The value of NPC's aggression level is divided by 3, the maximum possible level, to get a ratio representing their innate offensive capabilities. This part of the calculation acknowledges that a person's evaluation of a threat is not solely dependent on rational observations. The more aggressive the observing NPC is, the more favourable their perception of their ability to control the situation becomes. This portion of the equation was given a weight of 0.3 (30%) because a person with little aggression does not necessarily have less control over a situation than someone who is very aggressive. It simply means that they will realize that they will not always have complete control over external factors.

$$AttackProspect \in [0, 0.3] = 0.3 \times \left(\frac{SelfAggressionLevel}{3} \right) \tag{7.3.6}$$

The resulting calculation of *Attack Value* gives the observing NPC a rough estimation of their ability to control a threat via combat if it becomes necessary. It is maximized when the observing NPC's weapon power and aggression level are maximized and the observed NPC's defence power is zero. The minimum value is achieved when the observing NPC's aggression is minimized

[28] Unaggressive (0), Aggressive (1), Very Aggressive (2), Frenzied (3)

and the observed NPC's defence is maximized. The minimum value can be achieved regardless of how strong the observing NPC's weapon power is because the scaling factor uses a relative scale when comparing weapon ratings.

The second half of the *Threat Control* calculation, *Defence Value* (Equation 7.3.7), estimates how well the observing NPC can defend themselves from any potential combat threat posed by the observed NPC. Similar to *Attack Value*, the first component of the calculation creates a ratio between the observed NPC's weapon power $\in$ [4,32] and the largest of either the observing or the observed NPC's weapon power, which is used to scale the *Defence Value* based on the perceived strength of the observed NPC compared to the observing NPC.

$$DefenceValue \in [0,1] = D \times DefenceRating \times DefenceProspect$$

where

$$D = \frac{OpponentWeaponPower}{\max(SelfWeaponPower, OpponentWeaponPower)} \tag{7.3.7}$$

The calculation for *Defence Rating* (Equation 7.3.8) is similar to the calculation of *Attack Rating*, but this time it evaluates the observing NPC's defence as a ratio between their own defence power $\in$ [0,567] and the largest of either their own or the observed NPC's defensive power. This value is treated in the same manner as *Attack Rating*. The difference is that, because the parent equation *Defence Value* is subtracted from *Threat Control*, large values are achieved when the observing NPC's own defences are weaker than their opponent's, resulting in an overall lower *Threat Control* evaluation. This is the expected result since some groups consider a weak defence detrimental to their ability to manage threats, especially since one cannot be the initial aggressor using defensive capabilities. With this in mind, the *Defence Rating* can be viewed as a measure of how impervious the observing NPC is to external factors, which could be considered a form of control.

$$DefenceRating \in [0,1] = 1 - \frac{SelfDefencePower}{\max(SelfDefencePower, OpponentDefencePower)} \tag{7.3.8}$$

Similar to the way that aggression is used in the *Attack Prospect* equation to humanize the final result of *Attack Value*, the pre-existing game attribute of confidence[29] is used to humanize the final result of *Defence Value* via the *Defence Prospect* (Equation 7.3.9). It is also scaled to the range of [0,1] by dividing it by 4, the maximum possible confidence level. However, unlike

[29]Cowardly (0), Cautious (1), Average (2), Brave (3), Foolhardy (4)

aggression, confidence can have crippling effects on a person's situational evaluations. This resulted in the decision to multiply its value with the other components of *Defence Value*, causing the final result of *Defence Value* to become one when the observing NPC's confidence is minimized and zero when it is maximized. This inverse relationship causes the overall *Threat Control* evaluation to become more negative with high *Defence Value* results.

$$DefenceProspect \in [0, 1] = 1 - \frac{SelfConfidenceLevel}{4} \tag{7.3.9}$$

The final result of the *Defence Value* calculation is inversely proportional to the NPC's perceived ability to defend themselves against combat-based threats. A lower result means that the observing NPC is better equipped to endure potential threats compared to a higher result. This was done so that a weak defence results in a negative perception of the NPC's ability to manage threats, lowering the result of *ThreatControl*. *Defence Value* is maximized when the observed NPC's weapon power is maximized and the observing NPC's defence power and confidence are minimized. The minimum *Defence Value* is achieved when the observing NPC has at least one of defence power or confidence maximized. The minimum value does not depend on the observed NPC's weapon power due to the scaling factor that makes a relative comparison between the observing and observed NPCs' weapon power.

An extension to these calculations would be to include a weapon's range such that the overall attack or defence calculations are discounted if a weapon cannot reach their target. This extension was not implemented so that it was more likely that changes in NPC behaviours could be observed, but it could be included in future implementations. Another extension that was considered was using the levels of the player and NPCs. Originally, this extension was implemented as part of the threat control equations, but an NPC's level rarely, if ever, changes while the player's level can grow infinitely. This implies that the *Threat Control* portion of *Controllability* will eventually always cause a negative bias, which could result in the loss of an NPC's ability to express half of the basic emotions[30]. Another observation to note for future development is that NPCs rarely, if ever, change their equipment or spells, an ability that the player can use freely. The player can exploit this ability to elicit difference responses from NPCs, aligning with the role-playing aspect of *The Elder Scrolls V: Skyrim*. Additional game mechanics could be designed to exploit this observation if GLaDOS-specific game tasks were created.

Since the values used in the threat control equations are tightly linked to *The Elder Scrolls V: Skyrim*, it is calculated externally in the `GSensoryInput` module after the observed NPC's `GProfile` module has been processed

[30]The dimension encodings for Joy, Trust, Disgust, and Anger all require a positive value for *Controllability*.

by the `GAttention` module. It is then passed into `GPrimaryAppraisal` for use. This ensures that the calculations are only performed when required and removes the need to alter the `GPrimaryAppraisal` module for different games.

The *Prediction Bias* factor (Equation 7.3.10) recognizes that there are additional influences that can affect a person's perception of control that are directly related to their predicted outcome of the scenario. Positive *Prediction Bias* values indicate a state of relaxation or relief such that the NPC predicts that the outcome will be favourable, whereas negative values indicate a state of anxiety or panic where the NPC predicts that the outcome will be unfavourable.

For this interpretation of situational prediction, three elements are created and tailored for each NPC: *Goal Responsibility* $\in$ [0,1], *Stimuli Familiarity* $\in$ [-1, 1], and *Item Modifiers* $\{i : i \in [-1, 1]\}$. Specific values for these elements are stored in the `GGoals` module and extracted during information conditioning (Section 7.3.3). *Goal Responsibility* quantifies how accountable the NPC feels for maintaining the associated goal. A low value means that the NPC feels that the goal's success has little to do with their actions, whereas a high vale means that they feel that the goal's success heavily depends on their actions. This factor is combined with *Stimuli Familiarity*, a representation of the NPC's impression of the stimuli which is an accumulation of past encounters, knowledge, and biases. This results in a biased estimation of how favourable the encounter's outcome will be. High *Goal Responsibility* values cause stronger reactions to *Stimuli Familiarity*. This combination was created to capture the observation that people become more relieved, and are therefore more likely to predict a favourable personal outcome, if a stimuli aids in the achievement or maintenance of goal they feel strongly responsible for, and more likely to predict an unfavourable personal outcome if the stimuli hinders these goal.

$$PredictionBias \in [-1, 1] = \frac{P}{1 + |\, ItemModifiers\,|}$$

where

$$P = (GoalResponsibility \times StimuliFamiliarity) + \sum ItemModifiers \quad (7.3.10)$$

An optional set of values, *Item Modifiers*, are used to represent the phenomenon where a person is affected by seeing a particular item, such as a favoured faction's crest or a "cursed" item. These values are added to *Prediction Bias* to simulate a compounding-style evaluation. In many cases, the number of items that are considered in this calculation is low by design. This means that it is rare for the modifier values to outweigh the *Goal Responsibility*

and *Stimuli Familiarity* factors, which are expected to be the main contributors to a predictive evaluation due to their implied mental conditioning of the observing NPC.

Calculating the Effort Dimension

The third dimension, *Effort*, is an abstract quantification of how much energy an NPC is willing to expend in response to a stimuli. Positive values mean that an NPC wants to expend energy, whereas negative values mean that an NPC is opposed to expending energy. In this abstraction, *Effort* is influenced by two factors: personality and the severity of the event encountered with respect to the predicted outcome and the affected goal's value.

Personality is represented by the NPC's *Energy Profile*[31], a GLaDOS-specific attribute styled after the confidence and aggression values already present in *The Elder Scrolls V: Skyrim*. It is used to denote how much of the calculated effort the NPC is willing to expend based on their personality or current mood. From observation, a lethargic personality is less likely to want to expend energy whereas a motivated personality is more likely to expend energy. To model this, the range of *Energy Profile* is shifted to [-2,2] such that the possible values are centred about zero. This ensures that low *Energy Profile* values are more likely to result in a negative result and high values are more likely to result in positive results. The *EnergyProfile* value is used to shift the *Prediction Bias* factor from the *Controllability* equation to determine whether or not an NPC is willing to expend energy to alter the outcome of the encounter. Before it is applied to *Prediction Bias*, the remapped value of *Energy Profile* is multiplied by $\frac{3}{4}$ to get the shift's magnitude. A map value of $\frac{3}{4}$ was chosen because it ensures that *Prediction Bias* is overwhelmed when *Energy Profile* is assigned an extreme value. Since extreme personalities are less likely to care about their perception of an encounter's outcome, it is reasonable to assume that their willingness to expend energy depends more on their personality than their foresight. The shifted value of *Prediction Bias* is amplified by the affected goal's value, represented by *Goal Utility* from the *Desirability* equation. Since *Goal Utility* is never negative, its value does not affect an NPC's willingness to expend energy. Instead, it determines how committed the NPC is to that choice. Compared to the range of the shifted *Prediction Bias*, $\left[-\frac{5}{2}, \frac{5}{2}\right]$, the impact of *Goal Utility* is scaled relative to *Energy Profile*, still ensuring that NPC personality is the main contributor to the *Effort* calculation.

The target range of *Effort* is [-1,1] so that it aligns with the other dimensions, *Desirability* and *Controllability*. In order to achieve this range, the final result must be multiplied by a factor of $\frac{2}{5}$ to counteract the effects of the

[31] Lethargic (0), Lazy (1), Average (2), Motivated (3), Go-Getter (4)

Prediction Bias shift.

$$\textit{Effort} \in [-1,1] = \frac{2}{5}\,\textit{GoalUtility} \times \left(\frac{3}{4}(\textit{EnergyProfile} - 2) + \textit{PredictionBias}\right) \tag{7.3.11}$$

Calculating Emotion Intensity and Calling the Emotion State

Once individual dimensions are calculated, they are matched with an emotion based on their mathematical signs (Table 7.1) to produce an emotion code and intensity $\in [0, \sqrt{3}]$ (Equation 7.3.12). As with the PAD model, a three-dimensional vector space was chosen so that the dimensions of *Desirability*, *Controllability*, and *Effort* can be represented as a vector. This allows the vector's magnitude to be used as a measure of emotion intensity. Due to their design, an axis represented by any of these dimensions has an absolute magnitude of one in any given direction. The same approach was used in ALMA (Gebhard, 2005) as part of their emotion strength mapping process.

$$\textit{Intensity} \in [0, \sqrt{3}] = \sqrt{\textit{Desirability}^2 + \textit{Controllability}^2 + \textit{Effort}^2} \tag{7.3.12}$$

After the intensity of the emotion vector is calculated, it is scaled to the range of [0,100] $\in \mathbb{Z}$ in the main control module (`GLorthiem`). This is the intensity range of the `GEmotionCombinator` module, a child of the `GEmotionStore`, in the Emotion State. The resulting emotion code and scaled intensity are passed to the Emotion State to determine its effects on the current emotion values.

7.4 Emotion State

The main task of the Emotion State (Figure 7.5) is to condition and store the output from the Process Unit. It also acts as the bridge between the data analysis tasks of the Process Unit and selection and execution of NPC actions coordinated by Behaviour Expression (Section 7.5). This enables the use of stimuli appraisals during the action selection process. This section of the GLaDOS implementation has been customized for each NPC personality by assigning different emotion decay rates and equilibrium states in the `GEmotionDecay` module (Section 7.4.3).

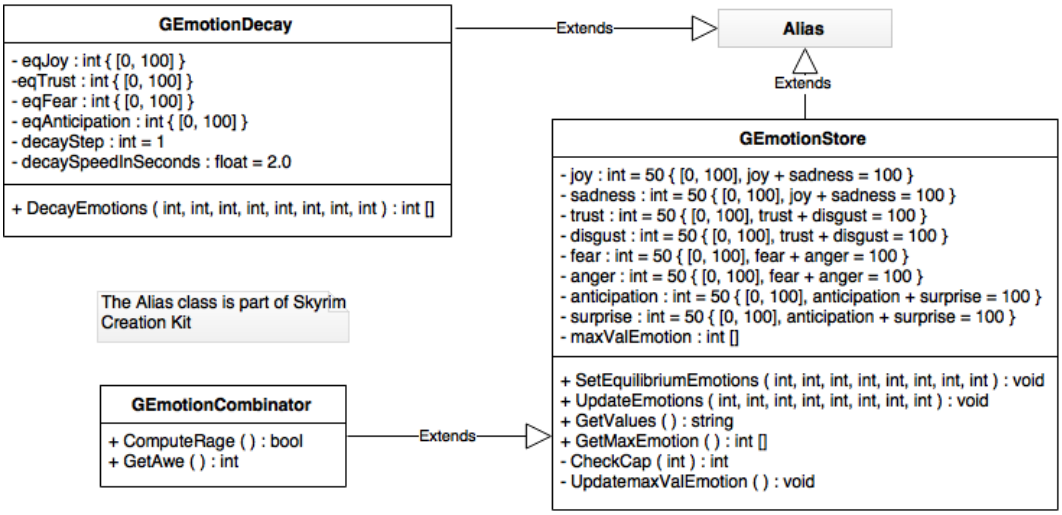


Figure 7.5: Emotion State Classes

7.4.1 Storing Emotion Values (`GEmotionStore`)

The core module of the Emotion State, `GEmotionStore`, stores and maintains emotion using the organization outlined in Plutchik's psycho-evolutionary synthesis. Pairs of emotions are stored as a set of percentages, individually $\in$ [0,100], where their sum must always be exactly 100%. This ensures that the pair remains balanced with respect to Plutchik's taxonomy.

When a new value arrives, the changes that it would make to the existing emotion intensity value is analysed to determine if an equal and opposite change to the opposing value in the pair has also been provided. This validates the incoming information, preventing an oscillation effect where the paired emotion intensities continuously swap their values because they are unable to achieve equilibrium (Section 7.4.3). If the validation fails, the difference of the incoming intensity values for a pair is used to update the stored emotion intensities instead. After they are updated, if the sum of the resulting paired values are $\notin$ [0,100], they are clipped at either 0 or 100. The boundary value selected depends on which one is exceeded.

7.4.2 Defining Complex Emotions (`GEmotionCombinator`)

In Plutchik's taxonomy, complex emotions are defined as combinations of basic emotions and intensity levels. This implies that they can be functionally defined as an extension of the basic emotion pair definitions. To facilitate this, the `GEmotionCombinator` module was implemented as child class of the `GEmotionStore` module.

Even though it is not explicitly used in this implementation of the GLaDOS system, the `GEmotionCombinator` is referenced by the main control class, `GLorthiem`, whenever functions from `GEmotionStore` are used. This eliminates the need to alter the control class or the underlying `GEmotionStore` module when the functionality provided by the `GEmotionCombinator` module is required, making the addition of user-defined augmentations easier to manage.

7.4.3 Defining and Enforcing an Equilibrium Emotion State (`GEmotionDecay`)

From observation, people tend to have a "base" emotional state or disposition in which they operate until they are aroused by a stimulus. When the stimulus is removed, either by removing the stimulus or moving to a new location, their emotional state returns to this baseline state. This presents a different set of functional requirements from the basic maintenance and storage of emotional values, resulting in the definition of a new module, `GEmotionDecay`.

The default state of the GLaDOS implementation sets an equilibrium value of 50% for all emotion intensities which is used to initialize the values in the `GEmotionStore` module. This default state definition eliminates some of the configuration tasks required to use the GLaDOS system. However, this definition might not be appropriate for all NPC designs and a game designer might want to selectively change certain NPCs' equilibrium values. To accommodate these situations, alternate equilibrium values can be used to configure the `GEmotionDecay` module. To ensure that user-defined equilibrium values conform to the intensity-pair constraint where their sum must be equal to 100%, only one element from each pair is visible in the equilibrium state configuration. Even though this does not allow game designers to directly alter some emotion equilibrium values, it is still possible to use the available variables to set the equilibrium state of their opposite. For example, if a designer wanted to define the equilibrium value of "Sadness" as 70, they would set the `eqJoy` variable, the opposing emotion of sadness, to 30 since $(100 - 70 = 30)$. During initialization, these alternate values are extracted and passed to the `GEmotionStore` to ensure that the NPC starts with their user-defined equilibrium state.

The second purpose of the `GEmotionDecay` module is to provide a means for returning to the equilibrium state if it is disturbed by an external stimulus. To facilitate this, the `GEmotionDecay` module has the ability to compare the values in the `GEmotionStore` module with its equilibrium state configuration. If the values are not the same, the module uses a decay step to move the `GEmotionStore` values closer to the equilibrium values. The rate at which this step occurs is measured in seconds and is dependent on the main control

class, `GLorthiem`, although the `GEmotionDecay` module contains a recommended decay rate which the control class can query for a smooth transition. The recommended decay rate can be altered during system configuration to suit individual NPC designs.

7.4.4 The Impact of Emotion State Changes on Behaviour Expression

Changes to the Emotion State do not immediately impact the GLaDOS system to allow for greater control over NPC behaviours. Instead, the Emotion State passes state change information to the main control class which then relays the data to the first Behaviour Expression module, `GSecondaryAppraisal`. In addition to giving more control to game designers regarding NPC behavioural changes, this method is resource-friendly because it only relays information when it becomes available rather than periodically checking for changes.

7.5 Behaviour Expression

In the GLaDOS system, the calculation of an NPC's emotion state has little value if it cannot be expressed. Methods for expressing different emotion states to the game environment, where the potential for the player to observe it exists, heavily depends on the game itself and how NPCs exist within it. However, the method for selecting which emotion to express can be made game agnostic. The Behavioural Expression unit (Figure 7.6) combines a game-agnostic selection mechanism with game-specific aspects of NPC emotion expression.

7.5.1 Selecting the Emotion to Express (`GSecondaryAppraisal`)

When the system wants to determine if an NPC should make an emotional expression, the `GSecondaryAppraisal` module is called. This module stores an ordered list of values representing expression thresholds for each of the possible emotions. The list is ordered to be able to implicitly control which emotions come more easily to different NPCs, and can be considered a form of personal expression. In the GLaDOS system, the value and ordering of each threshold was created for individual NPC personalities.

During the selection process, if an emotion intensity is equal to or exceeds the associated expression threshold, that emotion is encoded and returned to the main control class where it will be used to select an NPC behaviour and

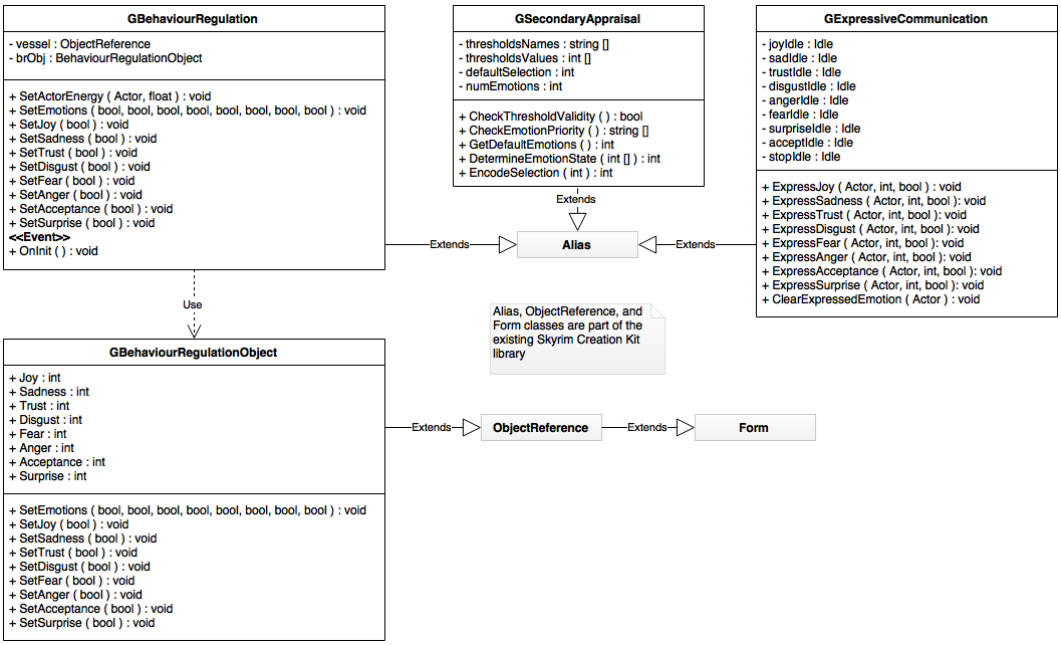


Figure 7.6: Behaviour Expression Classes

animation. If none of the emotion intensities match their expression threshold, a default emotion code is returned. This selection process concludes the game-agnostic processing capabilities of the GLaDOS design. The remaining elements are game-dependent because they interface directly with the game environment through the manifestation of NPC behaviours and animations.

7.5.2 Expressing Emotions – Active (`GBehaviourRegulation` and `GBehaviourRegulationObject`)

Emotion codes selected by the `GSecondaryAppraisal` module are used to enable two distinct expression types: behavioural and communicative. Behavioural expression refers to the functional AI-guided behaviours that an NPC uses, whereas communicative expression refers to cosmetic NPC animations. Communicative expression is not necessarily tied to a specific behaviour leading to its separation from behavioural expression. Unfortunately, the use of AI packages was limited to the built-in templates and pre-set behaviours because it is difficult to include user-defined NPC behaviours without introducing additional design overhead that is beyond the scope of the GLaDOS design.

In *The Elder Scrolls V: Skyrim*, functional NPC behaviours are defined in an AI Package[32], which an NPC can be set to use when specific conditions

[32]Creation Kit Wiki – AI Package Templates (Editor)

are met. Most NPCs have a default package defined which will be used if no other package conditions are met. In the GLaDOS implementation, this package was treated as the default emotion state behaviour so that emotion-specific packages could be triggered as required. An AI Package was created for each expressible emotion using the built-in templates, and the resulting package designs often contained simple behaviours that were similar to the NPC's default package. This was done because no other templates appeared to be more appropriate for an emotion's expression. The decision affects the basic emotions of Joy, Sadness, Trust, Disgust, Anticipation, and Surprise. The "fight or flight" emotions of Anger and Fear were assigned different package templates because there appeared to be reasonable alternatives such as a designated "Flee" template.

The emotion intensities from `GEmotionStore` is incorporated into the AI Packages by changing the NPC's Energy Level[33], which is used to dictate how quickly an NPC moves between available tasks. Even though this value is incorporated into all packages, this addition is especially important to the emotions with packages similar to the default one because it changes how frequency an NPC rotates between package actions. Even small changes like this can affect how an NPC is perceived by a player because it can reduce the number of obvious repetitions in NPC behaviours. Unaltered emotion intensities are used to set the package energy level for the emotions of Joy, Anger, Fear, and Surprise. Sadness uses an alternate calculation, $(100 - Intensity)$, because people in low-spirits tend to have lower energy levels. Since it is the opposing emotion in the pair, the intensity of Joy is used to set the energy level of the Sadness AI package to reduce the number of required calculations. The emotions of Trust, Disgust, and Anticipation use the NPC's energy level as defined by the host game. These emotions do not tend to create an outward showing of energy from the affected party. Instead, they cause an internal build-up that turns into a more outwardly expressive emotion if left unchecked. For these emotions, their display relies heavily on animations rather than behaviours (Section 7.5.3).

Custom selection conditions[34], defined in the `GBehaviourRegulation-Object` module, were made to ensure that a GLaDOS-specific AI Package is selected when a new emotion state is entered. The conditions are switches that can be OFF (0) or ON (1) and exactly one switch can be ON at any given time. To be identifiable by the Creation Kit as a selectable AI package condition, the `GBehaviourRegulationObject` module must be attached directly to an object in the game environment. Most NPCs exist explicitly in the game environment, so this limitation does not affect them. For the remaining NPCs that do not explicitly exist in the game environment due to their

[33]Creation Kit Wiki – Energy (AI Data)
[34]Creation Kit Wiki – GetVMQuestVariable (Condition Function)

involvement in quests[35], their `GBehaviourRegulationObject` class was attached to other objects in the game environment that could not be affected by the player, such as flour sacks and tables[36]. While this made it possible to access AI Package conditions for these NPCs, it is not possible to guarantee an association between the `GBehaviourRegulationObject` module and a specific NPC target. The proxy module `GBehaviourRegulation` was created to address this issue. The `GBehaviourRegulation` proxy module can be assigned to an NPC's ReferenceAlias and associated with an object that the `GBehaviourRegulationObject` module can be attached to. Changes to an NPC's selection conditions can then be propagated through the proxy module to the `GBehaviourRegulationObject` module, exposing the changes to the AI package selection mechanism. This ensures that condition changes can be detected and used at runtime to change an NPC's active AI Package.

7.5.3 Expressing Emotions – Passive (`GExpressiveCommunication`)

The second part of emotional expression is communication, the more cosmetic display of emotion. As with AI Packages, the creation of new NPC animations was beyond the scope of the GLaDOS design, which limited NPC animation choices to those that already existed in *The Elder Scrolls V: Skyrim*.

For this implementation, the communication of an emotion is conveyed through facial expressions and Idles[37], a type of animation in *The Elder Scrolls V: Skyrim* used when the player is not interacting with an NPC directly. An NPC's facial expression is set with the *SetExpressionOverride* function[38], which accepts an expression code and intensity value $\in$ [0,100]. Although three types of codes defined for this function[39], only the codes for Mood were considered because they can still be observed when the player is not interacting directly with the target NPC. Even though there are not a sufficient number of emotion codes to match each of the basic emotions in the GLaDOS system[40], emotions that could not be matched with an expression code could be reasonably expressed with a neutral expression. The intensity value required by the *SetExpressionOverride* function is set using the `GEmotionStore` intensity values. To ensure that the same starting state is used for all changes, an NPC's facial expression is cleared before a new one is set[41]. In a typical

[35]In Windhelm, this mainly affects the guards and the Stormcloak field commander.

[36]For NPCs that explicitly exist in the game environment, the `GBehaviourRegulationObject` is attached to their character model.

[37]Creation Kit Wiki – PlayIdle (Actor Script)

[38]Creation Kit Wiki – SetExpressionOverride (Actor Script)

[39]The facial expression types are Dialogue, Mood, and Combat.

[40]There were no expression codes that directly matched Trust and Anticipation.

[41]Creation Kit Wiki – ClearExpressionOverride (Actor Script)

game session, the player's viewpoint is rarely focused on any NPC's face and the transition to a neutral intermediary stage is fast enough such that the transformation will likely go unnoticed.

In addition to changing an NPC's facial expression, explicit animations were included that are played at the start of a new emotion selection. This was implemented to help differentiate emotion states that were assigned similar AI Package designs. Unlike facial expressions, this animation type is not played unless a new type of emotion state is entered. This avoids strange behaviours, like animation loops, from occurring when an emotion state is entered that differs from the previous state in intensity only. Animations were selected for an NPC based on their predefined personality. An exception was made for the Fear emotion state, where all NPCs were given the same animation because there did not appear to be any comparable options. Only animations that did not depend on additional objects were considered to ensure that an NPC was not limited by their equipment or surroundings. Due to their design, some animations played continuously until they were manually terminated. To counteract this, a stop animation[42] is always used before playing a new one to avoid awkward transition times. The use of NPC facial and Idle animations elevates otherwise similar AI package behaviours into distinct expressions of basic emotions, which should help prevent players from noticing repetitive NPC actions over extended periods of play.

7.5.4 What The Player Sees

The Behaviour Expression unit produces a compound NPC reaction using both functional and cosmetic NPC actions. Functional behavioural changes via AI Package selection demonstrate that an NPC's internal emotion state has changed, while cosmetic facial expressions and Idle animations help players determine what type of emotion an NPC is experiencing. If configured well, this human-like behaviour might lead to more believable NPCs. Initially, these changes might confuse a player who is unaware of the inclusion of the emotion-based system, but might prompt them to experiment to find out what caused the NPC behaviours. This experimentation could lead to a player's understanding of the NPCs themselves, which in turn might make the NPCs appear more believable. This results in a system that is more conducive to role-playing because players will begin to choose their equipment based on their location in the game environment or desired experience.

[42] A Stop Idle is used to make the transition to no animation appear more natural.

Figure 7.7: Classes for NPCs Created After System Initialization

7.6 Handling Non-Permanent NPCs

For simplicity, the GLaDOS Quest object completes system initialization when the game first starts. However, it is not always true that all NPCs have been added to the game environment at that time. An NPC's exclusion from the game environment can be caused by incomplete quests or other similar player-influenced factors. To handle the initialization process for affected NPCs, an extra class called `GAdelaisa` (Figure 7.7) was added to the GLaDOS system implementation. It enables the manual execution of initialization tasks defined in the `GLorthiem` control class for individual NPCs after the GLaDOS system is initialized. This class must be attached directly to target NPCs prior to running the GLaDOS system so that a reference to the desired NPC ReferenceAlias in the GLaDOS Quest object can be manually attached to the `GAdelaisa` object. Since Alias assignment only occurs when quests begin, an explicit reference to an NPC's ReferenceAlias object is required after the quests start. This enables the ability to re-evaluate alias assignments via scripted commands, forcing the GLaDOS system to be attached and initialized on NPCs that enter the game environment after the GLaDOS Quest object begins.

7.7 Conclusion

Implementing the GLaDOS system as a mod for *The Elder Scrolls V: Skyrim* posed many challenges. Even though the architecture was based on specific psychological theories, the theories did not contain sufficient information to ensure that strong implementation decisions could always be made. In order to complete a specification for the GLaDOS system implementation, information from additional sources was required, which included other psychological theories, implementation strategies from similar systems, and personal observation. The use of information from sources other than the GLaDOS design means that there are other possible implementations which future designers should consider when using the GLaDOS design in their game.

Due to the nature of the Creation Kit and its associated scripting language, Papyrus, several uncommon practices were employed to ensure that the

GLaDOS system could run in the host game. This included the creation of a Quest object and several ReferenceAlias objects for GLaDOS system initialization, which are designed to facilitate the creation of game challenges. Despite these workarounds, it was possible to implement the proposed GLaDOS architecture within *The Elder Scrolls V: Skyrim* with minimal deviations from the original design.

Implicitly included in the GLaDOS architecture, the `GLorthiem` control class is used to facilitate system initialization, track periodic updates, and coordinate inter-module communication. A helper module, `GAdelaisa`, was created to ensure that NPCs absent in the game environment during system initialization are correctly setup to use the GLaDOS system when they become available.

The Process Unit modules required the most deviation from the GLaDOS design. Structurally, its implementation required the addition of a detection system to determine when the GLaDOS system should be run, player-specific objects for NPCs to extract for analysis, and a data structure script to pass values for the appraisal calculations in bulk. The detection system was implemented to circumvent a known resource management problem in *The Elder Scrolls V: Skyrim* environment, where the game's native detection mechanism could cause data loss if used frequently. While the implemented detection system is more complex, its ability to run a series of simpler tests on a targeted NPC results in an overall improvement to resource management. The remaining structural additions are required by the `GSensoryInput` module so that it can create and store NPC information. It is rare for an NPC to change the information required for GLaDOS system calculations, making it less resource-intensive to create a data unit once instead of regathering the same information each time the GLaDOS system is used. Even though they are no different than the NPC versions, separate processing modules for generating and storing GLaDOS system information were created for the player. This was done to support future development such that a game designer can create separate game environment interfacing specifications for players and NPCs. The `GPrimaryAppraisal` module proved to be the most difficult part to implement because the psychological theories used in the architecture design only provided a general process of how information was converted into emotion values. To be able to fully specify appraisal calculations and an emotion encoding scheme, the GLaDOS implementation was forced to deviate from the design specification by including information that was not used in the GLaDOS specification. Additional information used to implement the `GPrimaryAppraisal` module included: the ALMA system to create a mapping between mathematical values and emotion categories and intensities; GAMYGDALA for its model of desirability; and patterns observed in the real-world and in psychology research. This deviation might be more desirable because the resulting

specification feels more like a traditional game design approach instead of an academically-sound, model-based approach. This could make more appealing to use, increasing the likelihood of acceptance in the game design community. There were some Process Unit modules that remained unaltered between the design and implementation stages because their tasks are relatively simple. Both the `GGoals` and `GAttention` modules were implemented as they were originally designed, filtering the data from an observed NPC's data profile and matching relevant information with the observing NPC's goal information.

The Emotion State proved to be the easiest section to implement because it does not interact directly with the game environment, and the psychological model supporting its design provided sufficient details to facilitate the translation to a computerized system. The `GEmotionStore` accepts incoming emotion encodings and intensities, rebalancing the internally stored values based on a balanced emotion pair structure; the `GEmotionCombinator` facilitates the combination of basic emotions and intensities into complex emotion definitions; and the `GEmotionDecay` module ensures that the internal emotion state is returned to its original state over time. Even though it was not utilized in this implementation, the `GEmotionCombinator` was implemented to demonstrate how it interacts with core system modules and to facilitate user-defined extensions to the GLaDOS system.

Similar to the Process Unit, Behaviour Expression was challenging to implement because it combined system-agnostic and system-dependant components. The `GSecondaryAppraisal` and `GExpressiveCommunication` modules were relatively simple and closely follow the intended design, but one could be implemented independently of the host game while the other could not. While the `GSecondaryAppraisal` only relies on the game-agnostic Emotion State design and could be implemented independently of the host game, the implementation of `GExpressiveCommunication` uses game-specific details because it is used to control NPC facial and Idle animations. Unfortunately, the `GBehaviourRegulation` module deviated from the GLaDOS design due to limitations in the Creation Kit. Like `GSensory-Input`, this module required a helper script to be able to interface with non-permanent NPCs. The helper script, `GBehaviourRegulationObject`, was attached to all NPCs that use the GLaDOS system for implementation consistency and organization. The `GBehaviourRegulationObject` script allowed for the definition of conditional flags which enable the use of multiple AI packages. The creation of AI packages allows NPCs to express their internal emotion state via functional behaviours. Even though the difference between some of the AI package designs and an NPC's default package is minimal, it is still possible for a player to notice and take interest in the cause when the functional behaviours and cosmetic animations are used together. The intensity of an NPC's internal emotion state is incorporated into AI packages via

the energy level parameter, which impacts the rate that the NPC alternates between package tasks.

The implementation effort produced a testable version of the GLaDOS system in *The Elder Scrolls V: Skyrim*. As expected, system modules that directly interface with the game environment required additional helper scripts to facilitate communication between the game environment and the GLaDOS system. The helper scripts aid tasks such as initialization functionality, NPC detection, information gathering, and functional emotion expressions via NPC behaviours. System components that did not directly communicate with the game environment did not deviate from their design and required no additional specifications during their implementation. The exception to this was the `GPrimaryAppraisal` module, where environmental information is converted into emotions. Due to the under specification of the primary appraisal process in the underlying psychological theories, additional implementation details were defined. This resulted in a functioning, but ill-supported, module. Future implementations of the GLaDOS system should be used to test different primary appraisal models with the goal of finding a specification that can be integrated into the underlying GLaDOS design. Despite its flaws, this implementation can still be used to test the GLaDOS system's effects on one factor that affects a game's replayability value – player's engagement.

Chapter 8

User Testing the GLaDOS System

In order to see if the proposed system impacts player experience, and a game's replayability value by extension, a small user study was conducted. Positive game experiences often result in an increased replayability value for that game because players want to continue the enjoyable experience. Even though the purpose of the GLaDOS system is to increase a game's replayability, its ability to meet this goal cannot be measured by this study alone because it is influenced by multiple factors. Instead, this study can indicate if the proposed system has the potential to influence replayability in the CRPG genre by extracting general patterns from participant feedback about their experience with the GLaDOS system. This will help determine if there is value in refining the GLaDOS system to maximize its ability to improve a game's replayability value.

Since the scope of player experience is broad, player engagement was selected as the study metric. However, there is no direct way to measure player engagement due to the subjectivity of human experience. Instead, player engagement indicators were used as a way to convert subjective experiences to objective data. In this study, time was used as the indicator. Typically, a highly engaged player will not know how long that they have been playing a game whereas a weakly engaged player will be aware of the passing time. Using this knowledge, engagement can be indirectly measured by observing how often players want to know what time it is. This study was designed to be similar to studies used during a commercial game's development to see if it appeals to members of the target market.

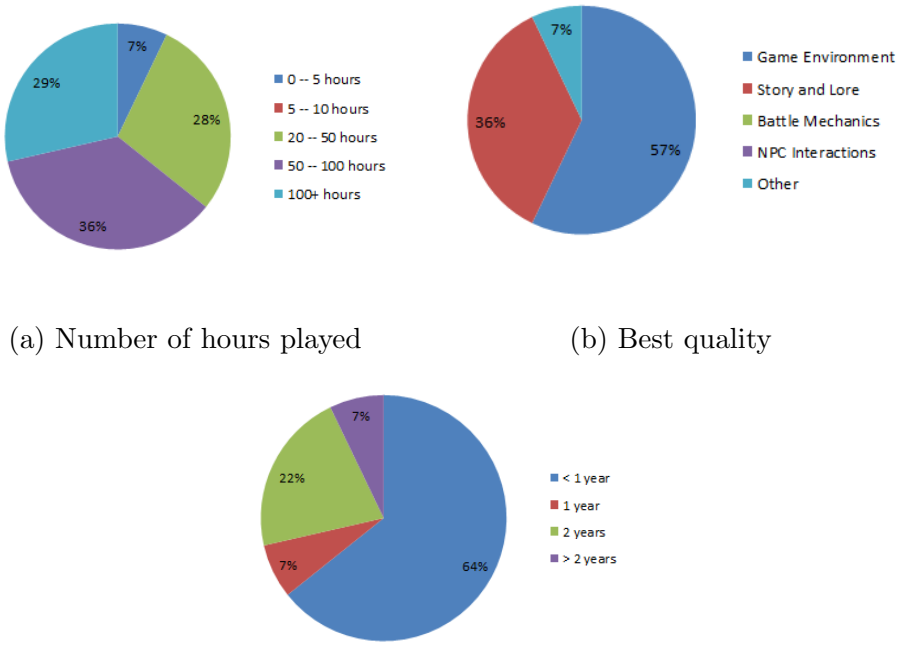
8.1 Participants

Fourteen people volunteered for the study of ages ranging from 19 to 26 ($M = 22.4$, $SD = 2.3$). All the participants were male and had normal or corrected-to-normal vision. They all had at least some prior experience playing *The Elder Scrolls V: Skyrim*, and additional information pertaining to their gaming habits was collected (Figure 8.1). This included approximately how many total hours they had already played the game and what they thought was the game's best quality. Most participants (36%) had played between 50 – 100 hours (Figure 8.1a) and selected the game environment (57%) as the best quality (Figure 8.1b).

Participants were also asked for an approximate date as to when they last played *The Elder Scrolls V: Skyrim* to help gauge how recent their experience was. The amount of elapsed time since their last play session can imply, along with the total number of hours played, how likely they are to notice differences between the original game and an altered version. It was essential to recruit participants who have at least some prior experience playing *The Elder Scrolls V: Skyrim*, even though this makes it more likely for participants to encounter situations that could lower their engagement with the game. One of the GLaDOS system design requirements was for it to be implemented in a way that fit within the host game's world lore, preventing a player from becoming disengaged due to semantically incorrect NPC behaviours. These behaviours, which would not necessarily appear unusual to new players, could negatively impact more experienced players because, over time, they develop mental models of how each NPC should behave based on their narrative role and presentation. Only one participant (7%) had not played the game in more than two years (Figure 8.1c), which could have negatively affected their feedback comparing the original version of game with the GLaDOS system version.

8.2 Apparatus

The study was conducted on a desktop PC with a 3.40GHz 4-core Intel Core i7 processor and 8GB of RAM. The operating system was Windows 10 Pro 64-bit. A 30inch Dell U3014 monitor (2560x1600 resolution with a 60Hz refresh rate) was used for all conditions. Participants could play the game with either a keyboard and mouse (Logitech K200 series) or a wired Microsoft Xbox 360 controller depending on their preference. They were also provided with a pair of over-ear stereo headphones (Beyerdynamic DT 990, 600Ω audiophile model) which were plugged directly into the PC via an audio plug. Participants were asked to sit at the PC as they would normally for the duration of the study (Figure 8.2).



(a) Number of hours played

(b) Best quality

(c) Most recent date played

Figure 8.1: Pre-experiment participant responses

For the control condition, the original release[1] of *The Elder Scrolls V: Skyrim* was run with no additional plug-ins installed, including the official downloadable content (DLC) plug-ins. For the system test, only the plug-in that contained the GLaDOS system implementation was added. Players were asked to use their own game save file for the study so that they could play using their own avatar. This required the use of Valve's Steam[2], a digital distribution platform for game management and social networking in the gaming community. The Steam platform features the ability to sync a player's game save files between multiple computers, allowing participants to access their game files from the PC used in this study. If a participant was unable to access their save file, one was assigned to them by the experimenter. The same save file was provided for all participants that required one and was initially selected at random from the files available to the experimenter.

[1]The special edition (*The Elder Scrolls V: Skyrim Special Edition*) had been released on October 25, 2016, making this distinction necessary.

[2]Valve Corporation's Steam installation page (http://store.steampowered.com/about/)

Figure 8.2: Hardware setup showing a participant playing *The Elder Scrolls V: Skyrim*

8.3 Procedure

Upon arrival, the experimenter explained the purpose of the study and asked for the participant's informed consent (Appendix C.2) before continuing. Participants were then asked to access their Steam platform accounts to ensure that their save file was available for use and that it did not rely on any plug-ins. If the compatibility of the participant's save file could not be confirmed, they were assigned the experimenter's save file. Participants were then asked to complete a pre-study questionnaire about their previous experience with *The Elder Scrolls V: Skyrim* (Appendix C.1.1). After the questionnaire was completed, the participant was given their choice of input device.

While the participant was not looking, the experimenter ensured that the first version of the game was running and loaded the selected save file. If necessary, the experimenter teleported the participant's avatar to the in-game city of Windhelm to ensure that participants were in the same game environment as the NPCs with the GLaDOS system installed. The participant was then allowed to begin their first play session, doing as they pleased within Windhelm, for 20 minutes. After this, the participant was stopped and the experimenter switched to the second version of the game while the participant was not looking. The participant then completed another 20 minute game session. During the play sessions, the experimenter wrote down the times at which the participant checked a clock that was set-up to the side of the PC.

Once both play sessions ended, the participant was asked to complete a post-play questionnaire about the two versions and leave any additional feedback that they might have (Appendix C.1.2). This concluded their participation in the study. For the security of their personal Steam account, the

experimenter ensured that the participant was successfully disconnected from the Steam platform before the participant left the study room.

8.4 Design

This study used a within-subjects design. The independent variable was version (original and system) which were trivially counterbalanced by alternating which version participants played first. The study was designed as a blind test, meaning that participants were unaware of which version they were playing at any given time. Participants were allowed to play each version for 20 minutes, yielding twenty-eight 20 minute trials. Trials were limited to 20 minutes so that participants could complete the study in one hour. This constraint could have artificially altered the engagement indicators, which are time-sensitive, by not allowing participants to play continuously until they stopped on their own. However, a continuous play-style design would require more participants such that each one would only need to play one version of the game. This would prevent participants from exhausting themselves on the first version of the game, leaving them unwilling to play the second version for very long.

Dependant variables tracked during play sessions include the amount of elapsed time before the participant first checked the time (FCHECK) and the average amount of elapsed time between checks (ECHECK). Since the data used to specify these values was manually collected by the experimenter, there is potential for errors during data collection. Although it was not used in this study, video recordings of participants might be useful for managing this type of error. Recordings would allow the experimenter to revisit participant sessions and record checkpoints at their own pace. Having copies of participant sessions would also allow for additional analysis, such as participant facial expressions and reactions. This would allow an experimenter to collect different types of data depending on their goal, such as the identification of GLaDOS behaviours that could be improved.

The detection system can have a significant impact on player experience because it dictates how frequently an NPC can be marked for GLaDOS system processing. Due to its periodic design, the selection of detection period lengths should be made to minimize blind spots in the system caused by the detection field's sleep period. For this study, the detection system was assigned a one second active period and a one second sleep period.

8.5 Analysis

Since the samples for the original and system conditions were not collected from independent groups and the calculated differences between the conditions

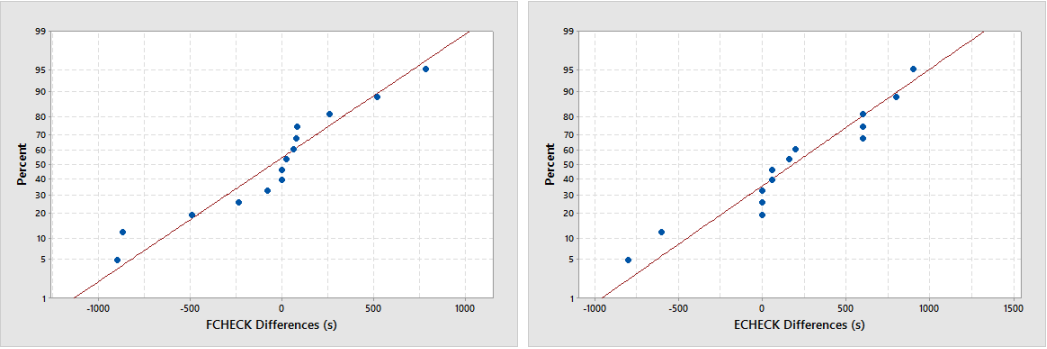
for both FCHECK (M = 55.0s, SD = 463.0s) and ECHECK (M = -184.0s, SD = 490.0s) appear to follow a normal distribution, a paired t-test was chosen to check for significance.

The assumption of normality for the differences in paired samples was checked using a Shapiro-Wilk normality test. The test accepted the null hypothesis that the difference values for FCHECK ($SW = 0.929$, $p > 0.05$) and ECHECK ($SW = 0.924$, $p > 0.05$) follow a normal distribution. A visual inspection of the probability plots for the differences between the variable conditions suggest that the data does follow the line-of-best fit (Figure 8.3).

There was no significant difference in the means for FCHECK ($t_{13} = 0.45$, $p > 0.05$, $d = 0.159$) between the original version of the game (M = 335.1s, SD = 327.0s) and the GLaDOS system version (M = 279.8s, SD = 369.1s) so the null hypothesis could not be rejected. This suggests that the player's initial engagement during the study was not influenced by which version of *The Elder Scrolls V: Skyrim* that they were playing.

There was no significant difference in the means for ECHECK (t_{13} = -1.41, $p > 0.05$, $d = 0.471$) between the original version of the game (M = 644.0s, SD = 389.0s) and the GLaDOS system version (M = 829.0s, SD = 397.0s) so the null hypothesis could not be rejected. This suggests that maintaining the player's engagement during the study was not influenced by which version of the game that they were playing.

Most participants reported that they noticed differences between the two versions of the game (85.7%) and were asked to record what those differences were (Section 8.6.1). Of the participants that noticed differences, most were able to select which version was the GLaDOS system (83.3%) and most preferred the version with the GLaDOS system installed (58%, Figure 8.4).



(a) FCHECK (b) ECHECK

Figure 8.3: Normal Plots for Dependant Variables

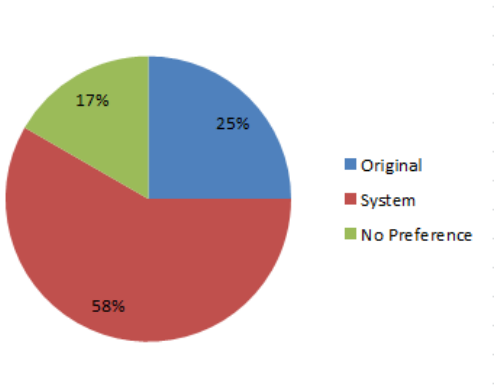


Figure 8.4: Participant Preference for Test Condition

8.6 Discussion

The objective analysis for both the initial and long-term engagement indicators could not determine if there was any difference between the original and GLaDOS system versions. While it is possible that there are no differences between the two versions, this cannot be determined by statistical analysis either making the analysis results inconclusive. Failure to detect differences could be attributed to a number of factors. Factors associated with the study itself include having too few participants and errors made during the data collection process.

There did not appear to be a predictable time in which participants first looked at the clock, nor were there only a few set counts that could be accurately chosen to represent the frequency at which they checked the time. This lead to a wider deviation from the averages, negatively impacting the ability to draw any significant test results from the data. This problem could be alleviated by recruiting more participants such that the variation in the data sets would begin to converge on a singular data point. The number of participants required to reach this point, however, would be unrealistically large so this might not be the best way to improve the results of a statistical test.

The other study-related factor, data collection errors, could be improved through the use of eye-tracking equipment. This would allow data to be collected automatically and reliably by recording times that the participant's eyes were no longer focused on the screen. This would also assist in grey areas during manual data collection where it is difficult to tell if a participant is looking off-screen or at an extreme corner of it. This was an issue during the study because some information in *The Elder Scrolls V: Skyrim* is presented in the top-right corner of the screen and the clock setup for the study was located on the right-hand side of the study area. The first improvement made to the

execution of this study is to data collection methods because the recruitment of additional participants is not a worthwhile effort until data can be reliably collected from them.

There were also a number of factors associated with the GLaDOS system itself that could have negatively impacted the statistical test's ability to detect differences. One possibility is the strength of the NPCs' reaction, which depends on the player avatar's race and current equipment. If the NPC calculated an emotion value that does not trigger a change in behaviour, they would not react to the player. This could cause the player to believe that the NPCs do not have that ability. While this could be alleviated by changing equipment, most participants did not because it was not apparent that it was the way to activate the GLaDOS system. One participant did figure out that equipment is the key to the GLaDOS system and began to deliberately change their avatar's equipment to elicit the responses that they wanted from NPCs. This suggests that if participants knew how the GLaDOS system worked, they would take advantage of it. The GLaDOS system had the additional limitation of not being able to poll the game environment at all times due to the design of the detection system. This limitation became a detriment when participants passed through an area too quickly, making it likely that some NPCs were unaffected when they would otherwise process a state change. Again, this presented the appearance that NPCs were unable to react to the participant's avatar.

Another possibility that could have affected a participant's engagement indicators was the in-game time at the beginning of their session. In *The Elder Scrolls V: Skyrim*, NPCs are assigned schedules in their AI packages dictating where they are supposed to be at any given time. At night, most NPCs are sent to their homes to sleep instead of wandering around the city which forces players to seek them out. For many participants, the initial in-game time was set to night time so there were not many NPCs for them to encounter. This could have adversely affected their initial engagement because there is little to do in the game environment selected for the study that does not involve NPC interactions.

Finally, the participants' own expectations of the GLaDOS system might have affected their initial engagement if they were not met. Expectations include what they expect will trigger NPC reactions and how NPCs should behave according to an NPC's personality and role within the game. Many participants noted that they were not very familiar with the Windhelm game environment because they did not have to go there while they were playing *The Elder Scrolls V: Skyrim*. One participant suggested that another in-game city, Whiterun, would have been a better testing environment because it is the first city that most players encounter when they start a new game. While relocating the test location from Windhelm to Whiterun might produce different results

since it is more likely that participants are familiar with the NPCs, it is unclear how much impact this switch would have considering that most participants were still able to accurately select which of the versions had the GLaDOS system installed and tended to prefer it over the original game.

8.6.1 Participant Feedback (Subjective Data)

While it cannot be determined at this time if there are differences in implied engagement between the original and GLaDOS system versions of *The Elder Scrolls V: Skyrim*, participants often knowingly and with the ability to refrain from choosing a preference selected the GLaDOS system as their preferred version.

Some participants felt that the NPCs took a more active role to engage the player, such as responding to threats more quickly, choosing their fights based on the player's equipment or perceived faction alliances, being more blatant in their treatment of the player based on their avatar's race, and interacting more frequently with the game environment. These behaviours were deliberately encoded in the GLaDOS system. Participants stated that this caused them to be more aware of their surroundings in case an NPC approached them and appeared to make the game harder. This is ideal for many players because they are often looking for increasingly difficult challenges the more that they play a specific game or genre. A specific behaviour that a participant noticed, guards taking more notice of dead NPCs and being more intolerant of crimes, was not part of the design but its perception might have been brought on by other intended effects.

It is not possible for the GLaDOS system to select an "incorrect" behaviour as a result of its processing because there were no designated "correct" behaviours. However, it is possible for the manifestation of behaviours to appear erroneous due to the design of the behaviour itself, which could sometimes result in unintended NPC behaviours. For example, a behaviour that was encoded into the NPCs' *Disgust* emotion, ignoring the player, was not always perceived as being deliberate, causing participants to perceive the behaviours as being unpredictable. Some of these unintended behaviours were benign and could be overlooked, such as fewer interruptions by NPCs during conversations, NPCs that did not stop what they were doing during player conversations, and NPCs that made fewer comments when players passed by. Other unintended behaviours that participants noticed, such as NPCs running away from their avatar despite their narrative role, were jarring and caused participants to think that an error had occurred in the game. Even though the fleeing behaviour is part of an NPC's expression of *Fear*, the behaviour was perceived as erroneous because the reaction did not appear to match the narrative role of the affected NPC. This unintended effect can be attributed

to implementation time constraints and a lack of familiarity with the Creation Kit, which resulted in little or no variation in the AI Packages used for each NPC. This caused near-identical emotional displays from NPCs that have different roles within the game's narrative. With the current AI Package designs, an average citizen and the field commander of a rebel army both attempt to run away from the player if they become fearful of them. While this would not be considered unusual behaviour for the citizen character, running away from a fearful situation is questionable for an army field commander and might be identified as desertion. This observation demonstrates the importance of creating behaviours and animations that make sense for the NPC's narrative role instead of using a generic design for all NPCs.

There were also some technical issues that likely impacted participant perceptions of the study. Due to the way that some game-related quests are setup, some NPCs are not physically present in the game environment until runtime. This means that their ReferenceAlias objects containing the GLaDOS system need to be attached at runtime as well. While it is feasible to do so, there is no guarantee that the alias will be attached properly, resulting in improperly initialized NPCs. In this case, the affected NPCs would default to their unmodified behaviours, which could appear strange when all other NPCs are working properly. One participant explicitly noted this error because the affected NPCs appeared strange when compared to properly initialized ones. Unfortunately there does not appear to be a reliable way to prevent this error in *The Elder Scrolls V: Skyrim* Creation Kit, so it is recommended that another host game should be employed for further testing. Some participants also noticed small frame rate changes between the original and GLaDOS system versions. It is known that *The Elder Scrolls V: Skyrim* is not a well-optimized game, making it difficult to add computationally-intensive tasks without impacting the game's performance. This technical limitation also supports the suggestion that further testing of the GLaDOS system should be performed in another host game. Unrelated to *The Elder Scrolls V: Skyrim*, the monitor used during the study was sometimes tinted blue for unknown reasons. Some participants found this distracting, which could have negatively impacting their engagement indicators.

8.6.2 Eliminating the Novelty Bias

At this time, it cannot be known if the subjective participant favour towards the GLaDOS system is due to its novelty or because it is the superior version. A longitudinal study, with some of the noted alterations to the testing procedure, testing apparatus, and GLaDOS system implementation, should be conducted to see if participant responses change over the course of multiple game sessions. If the preference for the GLaDOS system remains consistent,

there is a stronger indication that it is not because it is a novelty whereas a preference change indicates that the GLaDOS system is simply a new and participants are interested in seeing what it can do. A longitudinal study also offers the opportunity to collect additional objective data to improve the chances of statistical significance detection. The use of statistical tests is a more reliable and replicable method for comparing the differences between the original game and GLaDOS system when compared to the collection and analysis of subjective participant responses.

8.7 Conclusion

To test the viability of the GLaDOS system implementation in *The Elder Scrolls V: Skyrim*, a user study was conducted to determine if there were statistically significant differences in engagement indicators between the original and GLaDOS system versions of the game. Since replayability cannot be measured directly, player engagement was used as the study's metric to determine if the GLaDOS system had the potential to impact the more abstract concept of replayability. Engagement indicators were defined as the amount of elapsed time before participants first checked the time and the average amount of time between each check. Subjective data was also collected after participants played both versions of the game regarding their preferred version and what factors influenced their selection.

Comparing the objective data collected by observing participants and the subjective participant responses post-study, there does not appear to be any agreement regarding a superior version. A number of improvements have been proposed to improve the statistical significance of the objective data. Data collection improvements can be made by recruiting more participants for better statistical significance detection, and by collecting data more reliably and accurately through the use of an eye tracking apparatus. Improvements could also be made in the GLaDOS system itself by exaggerating NPC reactions to make them more noticeable; making it more apparent to participants how to trigger the GLaDOS system in NPCs; and by implementing an always-on polling system for scanning the game environment. Certain study conditions, such as the in-game time, should also be considered in future studies because it can influence how frequently participants encounter NPCs. Perhaps the biggest factor that affected the results of the study was participant expectations of the system, including what they thought would cause NPC reactions and what type of behaviours an NPC should exhibit based on their purpose in the game world.

There were also factors associated with the GLaDOS system implementation itself that impacted the subjective participant responses. Many intentional GLaDOS system behaviours caused participants to become more aware

of their surroundings because they felt that NPCs were actively seeking them out. Participants also felt that NPCs were more concerned with their self-preservation than in the original game because they were more selective when confronting the participants' avatar. These types of behaviours produced perception effects such that other behaviours that were not controlled by the GLaDOS system appeared to be intentional, increasing the NPCs' overall believability. While these types of participant responses are promising, there were also several comments about GLaDOS system behaviours that made NPC behaviours look erroneous. This observation was typically made when an NPC behaviour did not match the affected NPC's narrative role or when it was not obvious that it was a reaction to the participant's avatar. This indicates that even if the GLaDOS system is selecting the correct set of behaviours, the behaviours themselves have a large impact on player engagement and should be refined in future iterations. Additional technical factors that impacted the results of the study should also be considered, such as what game to implement the GLaDOS system in.

There are many improvements to make before another study can be conducted, such as improving data collection methods and tailoring NPC behaviours to their role and personality. It is strongly suggested that a longitudinal study is conducted to determine if the long-term participant preferences remain constant, after refinements to the GLaDOS system are made. This will better indicate if the current interest in the GLaDOS system is due to its novelty or if it has the potential to maintain a participant's interest over an extended period of time.

Chapter 9

Conclusion

The replayability value of a game is partially influenced by a player's perception of differences between consecutive playthroughs, which make them appear like a different game. Part of the challenge of game design is finding the point at which players can perceive these differences while limiting the number of additional implementation resources. The type and scope of variations used is genre dependent due to differences in complexity, challenge, and design purpose. This makes it unrealistic for a single design variation scheme to be used for all genres well. To limit the scope of the GLaDOS system design, the CRPG genre was selected because it focuses on role-playing within the game environment. This inherently demands some amount of replayability due to the variety of ways that players can present themselves. Interactions with NPCs is one of the main ways that a player can interact with the game environment in CRPGs, acting as intermediaries for a market system, specific quest roles, and as combat opponents. Despite their non-trivial presence in most CRPGs, NPCs often have limited capabilities to enhance a player's role-playing experience. This is due to the under specification of NPC agency which results in the creation of NPCs that pay little attention to the player outside of pre-determined contexts, which negatively affects their believability. The GLaDOS system was designed to provide game designers with a framework for creating higher agency NPCs. Human behaviours are influenced by a variety of factors, and attempting to include all of them in a single design is unreasonable at this time. To further limit the scope of the GLaDOS design, human emotion was selected as the underlying model.

Techniques for creating realistic NPC behaviours have already been created in both industry and academia. In industry, a combination of ad hoc, heuristics, and algorithms have been employed by designers to create increasingly complex NPC behaviours. While these approaches can be used in most games, each behaviour design tends to be game-specific with limited external applicability. To address this limitation, portability was included in the GLaDOS

system specification. In academia, there have been several attempts to create a general, computerized simulation of human emotion. Many of them, such as PARRY, the affective reasoner, and ACRES, were created to test proposed psychological models of emotion theory. While these systems are not ideal for video games due to their complexity, they do propose specific software architecture components, such as input filtering, which should be included in the final design. Although a variety of theories were employed for research-focused designs, systems designed specifically for video games, such as ALMA, GAMYGDALA, and Tok, tended to use the OCC and PAD space models without indicating that those models were selected after analysing other models. Despite using the same underlying psychological model, each system created a unique design that produced promising results. While the GLaDOS system was designed for the same purpose as these systems, using the OCC and PAD space models for the same reasons did not appear to be reasonable.

As part of the foundational research behind the GLaDOS design, the three classes of emotion theory from psychology were compared by analysing two well-known theories from each. This analysis was used to determine which theory, if any, were potential candidates for the GLaDOS design. After analysis, it was decided that no single theory was sufficient for a complete GLaDOS specification because each one only provided enough information to define a portion of the emotion creation process well. Therefore, two theories were used in the GLaDOS design. Lazarus' cognitive appraisal, which did not define emotion categories consistently, was used to specify a process for converting environmental stimuli into emotion values and the selection mechanisms for emotion expression. Plutchik's psycho-evolutionary synthesis was used to specify basic emotion categories, maintenance functions, and guidelines for combining basic emotions into complex ones as needed. This theory proposed a process similar to cognitive appraisal, but provided fewer functional details.

To help guide the creation of the GLaDOS architecture, it was decided that it was more important for it to be easily understood and usable by game designers at the expense of a technically correct implementation of the selected psychological theories. This lead to the formulation of three distinct architecture units tasked with a simplified version of the associated psychological components. The Process Unit handles the gathering and conditioning of system inputs, determining which ones are important for processing, and performs the initial conversion of game stimuli into emotion codes and intensities. Once an emotion code and intensity is created, the Emotion State stores and maintains them. Finally, Behaviour Expression extracts the stored emotion values from the Emotion State to determine which, if any, to express. The expression process includes the selection of an appropriate set of behaviours and animations to convey an NPC's internal state. Combined, these units create a

framework for implementing cognitive appraisal and psycho-evolutionary synthesis without being tied to theory-specific details which is likely unimportant to a game designer's task.

In order to gauge player reception of the GLaDOS architecture, a basic version was implemented for the PC game *The Elder Scrolls V: Skyrim*. Even though this game was initially selected because it comes with tools for developing user-generated content, *The Elder Scrolls V: Skyrim* is a good benchmark game for the applicability of the GLaDOS system to other CRPGs. Since it is a sandbox CRPG, this game has an extended scope of player choice. A favourable player reception of this implementation indicates a high chance of acceptability in other CRPG types.

Like other academic systems for video games, PAD space was used as part of the implementation strategy because there were no known alternatives for the synthesis of stimuli into emotions at the time. This model still required alteration for simplicity and compatibility with Plutchik's psycho-evolutionary synthesis. As with similar systems that used PAD space, it cannot be said that using this theory in a computerized model is the best approach to use due to the lack of supporting evidence. Additionally, the use of this model is game-dependent because one of its calculations requires game-specific values. Despite these problematic design decisions, there was a larger implementation challenge concerning the NPCs' ability to express the basic emotions in the GLaDOS system due to limitations in the host game. Since this implementation of the GLaDOS system extends and existing game, the range of available emotional expressions is limited to pre-existing behaviour designs and animations. This led to sub-optimal behaviour expression designs, which could have adversely affected player perceptions. Polling the game environment for GLaDOS system inputs was also limited by the host game because it required the creation of a customized detection system that could not always be turned on. This sometimes caused NPCs that were close to the player to be passed over for processing if the player moved out of range before the system was triggered.

Once the GLaDOS system implementation was completed, a user study was conducted to determine if player engagement indicators and preference was affected. Statistical analysis of engagement indicators was inconclusive, but participants typically preferred the GLaDOS system over the original game. This feedback is promising because the inconclusiveness of the statistical tests could be attributed to a variety of factors including limitations in the host game and the GLaDOS behaviour designs, and unreliable data collection methods. As part of the feedback process, participants also suggested a number of improvements that they would like to see in future GLaDOS implementations, including NPC behaviours that better fit their narrative role, more

reliable indicators to differentiate GLaDOS selected behaviours from game errors, and using a more familiar in-game environment. If further testing is performed, these changes must be taken into consideration and a longitudinal study performed to determine if long-term interest in the GLaDOS system can be maintained.

9.1 Applications of the GLaDOS System

Despite the limited external applications of this GLaDOS system implementation due to its reliance on *The Elder Scrolls V: Skyrim*, the underlying principles can be used in a variety of ways. The most obvious application is to commercial game projects, where more believable NPCs can be used to enhance a game world and the player's interactions within it. With the GLaDOS system's potential to create a different experience each time it is played due to different game world interactions, a game's replayability value could be enhanced with this kind of extension. The GLaDOS system could also be redeveloped as a game mechanic. For example, a game challenge could be designed so that players must be aware of how they are presenting themselves in-game and adjust their behaviours accordingly to succeed. Most game companies need to innovate and create new game experiences in order to survive, and the GLaDOS system provides one suggestion for direction.

The GLaDOS system also contributes towards the development of a foundation for the digitization of human behaviour for computer science and software engineering. While the focus of many previous works has been on the computer's ability to reason about knowledge and relationships, the GLaDOS system explores other human behaviours and abilities for modelling and implementation in a digital environment. The GLaDOS system also demonstrates how models and theories from other academic fields can be integrated with software engineering practices. This can be used to encourage projects that could benefit from a technical perspective that would not traditionally consider it.

Even though the GLaDOS system implementation in *The Elder Scrolls V: Skyrim* itself has a limited scope of application, a well-developed system based on the GLaDOS architecture could have a variety of uses. Since it is an interdisciplinary project encompassing models and techniques from game design, psychology, and software engineering, there is a potential benefit to each field. It could be used to refresh a long-running game series or inspire new franchises in the game design industry. It can also add to the collective understanding of computerizing human behaviours and demonstrates how a project could benefit from an engineer's approach to problem solving. In its current state, the GLaDOS system cannot provide immediate benefits to the field of psychology. However, further development of the GLaDOS system

could prove beneficial to any number of applications that require the simulation of human emotions.

9.2 Future Work

There are several candidate areas for future development. Some are specific to the implementation in *The Elder Scrolls V: Skyrim*, such as the refinement of AI Package designs and Idle animation selections, although this has little value beyond this application. There are two directions that future work can take: continued system refinement or the exploration of alternate core models.

System refinement refers to any developments or testing to the GLaDOS system as it was proposed. One obvious omission in the GLaDOS system is the ability for NPCs to process observed behaviours. Unlike the selection of equipment and race, behaviours that players can engage in should also affect how NPCs react to them. Observable player behaviours that could be considered during emotion appraisal include withdrawing or sheathing weapons, moving items within the game environment, and undesirable social behaviours such as running across a tabletop. This improvement would likely require the creation of a database containing domain knowledge and rules, which could have negative effects on implementation complexity and performance. Aside from this improvement to the scope of GLaDOS system process, additional changes could potentially improve its portability and robustness with respect to its emotional displays.

Although the CRPG genre was selected for this iteration of the GLaDOS design, there is no strong argument against designing a similar system for other genres. A simpler GLaDOS system design might be sufficient for this task because CRPGs are usually the most intensive in terms of NPC interactions. This hypothesis can only be tested by building systems for a variety of video game genres and running a user studies similar to the one used during the GLaDOS system's development.

Performance-related developments include testing the portability of the GLaDOS system and optimizing its processes. If it can be demonstrated that the GLaDOS system can be used in other games with minimal adjustments to the core design while maintaining game performance, it is more likely to be adopted for academic research and commercial applications. This could overlap with the testing of similar designs for other genres.

If the portability and performance of the GLaDOS system can be verified, the next improvement that could be made is the implementation of the augmenting components suggested in the design specification (Section 6.3.4). These components add functionality to the GLaDOS system which could potentially make each NPC feel unique via elements such as personality, mood,

and memories. While these elements affect the internal processing of individual NPCs, the social interaction and learning augmentations would facilitate a digital in-game community, improving NPC interactions within the social framework of the game world. Augmenting the GLaDOS system with simplified models of neurobiological systems would add a randomized element to NPC actions while still being grounded in biological models. Factors such as hormones are known to affect normal cognitive processing which result in unexpected behaviours. In extreme cases, this augmentation can be used to create an NPC with a mental illness which can cause a dramatic departure from known GLaDOS system behaviours.

Another broad category of continued research involves an examination of the underlying psychological models of the GLaDOS system. A good starting point for this type of continuation would be to conduct user studies comparing the GLaDOS system to similar systems, such as ALMA, Tok, and GAMYGDALA. Further research into psychological models of emotion would be guided by the studies' results to build a more robust software equivalent. This type of research would require several system design and user study iterations to incrementally build a system up from the best qualities of its predecessors. To avoid errors encountered during the testing of the GLaDOS system, the user studies should be run using a game created specifically for testing. This would also avoid tying the final system to a specific game.

Appendix A

List of Games

This appendix contains information regarding each game that is mentioned in this thesis. Note that the publisher and year of publication listed refers to the initial game release, which might differ between regions. More information about each game can be found by following the links in the game title.

Title	**Developer**	**Publisher**	**Year**	**Platforms**
Assassin's Creed: Brotherhood	Ubisoft Montreal	Ubisoft	2010	Xbox 360, PlayStation 3, Microsoft Windows, OSX, OnLive
Bioshock Infinite	Irrational Games	2K Games	2013	Xbox 360, PlayStation 3, Microsoft Windows, OSX, Linux
Danganronpa: Trigger Happy Havoc	Spike Chunsoft	Spike Chunsoft	2010	PlayStation Portable, PlayStation Vita, Microsoft Windows, OSX, Linux, Android, iOS

Continued on the next page

Continued from previous page

Title	Developer	Publisher	Year	Platforms
Danganronpa 2: Goodbye Despair	Spike Chunsoft	Spike Chunsoft	2012	PlayStation Portable, PlayStation Vita, Microsoft Windows
Façade	Procedural Arts	Procedural Arts	2005	Microsoft Windows, OSX
Final Fantasy VII	Square	Square	1997	PlayStation, PlayStation 4, Microsoft Windows, OSX, Android
Final Fantasy X	Square	Square	2001	PlayStation 2, PlayStation 3, PlayStation 4, PlayStation Vita
Gran Turisimo 5	Polyphony Digital	Sony Computer Entertainment	2010	PlayStation 3
L.A. Noire	Team Bondi	Rockstar Games	2011	Xbox 360, PlayStation 3, Microsoft Windows
Mass Effect	BioWare	Microsoft Game Studios	2007	Xbox 360, PlayStation 3, Microsoft Windows
Mass Effect 2	BioWare	Electronic Arts (EA)	2010	Xbox 360, PlayStation 3, Microsoft Windows
Pac-Man	Namco	Namco	1980	Arcade Cabinet
Path of Exile	Grinding Gear Games	Grinding Gear Games	2013	Microsoft Windows

Continued on the next page

Continued from previous page

Title	Developer	Publisher	Year	Platforms
Portal	Valve Corporation	Valve Corporation	2007	Microsoft Windows, PlayStation 3, Xbox 360, OSX, Linux, Android
Portal 2	Valve Corporation	Valve Corporation	2011	Microsoft Windows, PlayStation 3, Xbox 360, OSX, Linux
Saint's Row the Third	Volition	THQ	2011	Xbox 360, PlayStation 3, Microsoft Windows
Shin Megami Tensei: Persona 3	Atlus	Atlus	2006	PlayStation 2, PlayStation Portable
Shin Megami Tensei: Persona 4	Atlus	Atlus	2008	PlayStation 2, PlayStation Vita
The Elder Scrolls IV: Oblivion	Bethesda Game Studios	2K Games, Bethesda Softworks, Ubisoft	2006	Xbox 360, PlayStation 3, Microsoft Windows
The Elder Scrolls V: Skyrim	Bethesda Game Studios	Bethesda Softworks	2011	Xbox 360, PlayStation 3, Microsoft Windows
Tom Clancy's Splinter Cell: Chaos Theory	Ubisoft Montreal	Ubisoft	2005	Xbox, PlayStation 2, PlayStation 3, Nintendo GameCube, Nintendo DS, Nintendo 3DS, Microsoft Windows, Mobile, N-Gage

Continued on the next page

Continued from previous page

Title	**Developer**	**Publisher**	**Year**	**Platforms**
World of Warcraft	Blizzard Entertainment	Blizzard Entertainment	2004	Microsoft Windows, OSX
Xanadu: Dragon Slayer II	Nihon Falcom	Nihon Falcom	1985	Sega Saturn, Microsoft Windows, PC-8800 Series, PC-9800 Series, MSX, X1, FM-7

Appendix B

Software Requirements Document for the GLaDOS System

Note: This document was prepared using the Volere Requirements Specification Template. Sections that are not applicable for this project were omitted.

B.1 Project Drivers

B.1.1 The Purpose of the Project

The user business or background of the project

In recent years, video games have greatly improved the quality of their graphics and audio, while relatively few improvements have been made to AI, specifically NPC behaviours. Players have noticed this and often share their experiences with the "unrealistic" behaviours. It has been suggested that the next big advancement in video game quality will be more intelligent game AI (Graft, 2015). One type of behaviour that all people share is emotional reactions to events. It is possible that if this type of behaviour were to be included in the NPCs of a CRPG, players would be more engaged with the game and play it longer than they would have otherwise.

Goals of the Project

The goal of this product is to enhance player engagement and long-term interest in CRPGs by enabling game designers to create more realistic NPC reactions to player actions and in-game events via emotional reactions.

B.1.2 Stakeholders

The Client

Dr. Jacques Carette, McMaster University
As the thesis supervisor, Dr. Carette reviews and approves any research or design decisions made for this project. Any requirements or constraints that he provides must be accommodated, if feasible. Since this project is being developed as part of the thesis, Dr. Carette is also the customer.

Other Stakeholders

- Academic Integrity Office, McMaster University
  If any aspect of the project is deemed to be academically dishonest, the Academic Integrity Office will stop further project development.

- ZeniMax Media and Bethesda Softworks Legal
  Since this product will be designed as a mod for their game, *The Elder Scrolls V: Skyrim*, there is a requirement to ensure that the product is implemented within the scope of the Creation Kit End User License Agreement (EULA).

The Hands-On Users of the Product

- Players
  This user group is unconcerned with the internal workings of the system, but are very interested in observing how different system outputs are formed from different inputs. They want to be engaged and entertained while playing a game, so a system that hinders this goal should not be added. This group can help judge the final product by testing if the system interferes with their engagement or enhances it.

  There are a variety of reasons that players play games, but often it is for entertainment and fantasy fulfilment. If a game is done well, a player does not really know what else is going on around them in the real world. This engagement with the game is an important aspect of the player experience that should be maintained.

- Game Developers
  This user group will be the ones deploying this system in their game. They might also modify or extend the system to suit their design needs. Game designers will also likely be the ones maintaining and supporting the product after deployment. This group can help develop the product

by making suggestions as to the structure and order of system components, where likely extension points are, and what type of product documentation is required.

Game development studios are usually fast-paced, high stress environments and it is rare for a new technology to be adopted because each studio works differently. However, if a system is proved to be integrable with existing products, improves player enjoyment of the game, and is easy to understand and use, it is possible for new systems and techniques to be adopted for future projects.

B.2 Project Constraints

B.2.1 Mandated Constraints

Solution Constraints

Even though many aspects of human emotions are agreed upon by psychologists, there are still components which are subject to debate. For a computerized system, it is not feasible to adequately satisfy every proposed theory of human emotion because of their variation. Therefore, two theories have been selected for this product. The processing component of human emotion is based on Richard Lazarus's cognitive appraisal, while the emotion classifications and maintenance are based on Robert Plutchik's psycho-evolutionary synthesis. Cognitive appraisal was selected because, in addition to being held in high regard within the psychology community, it can be compared to computerized processes. Psycho-evolutionary synthesis was selected for emotion categorization and maintenance because it proposes eight basic emotions arranged in four groups of opposing pairs, and includes a framework for defining complex emotions as combinations of basic emotions and emotional intensities.

Implementation Environment of the Current System

This product will be implemented as a mod, which means that it is subject to all the constraints of its parent application, *The Elder Scrolls V: Skyrim*. Since this game is comparatively large and runs in real-time, this has implications for performance constraints.

Partner or Collaborative Applications

- *The Elder Scrolls V: Skyrim* is a large, open-world CRPG with an emphasis on player choice. This potentially leads to scenarios where several events occur simultaneously, which this product will need to process in order to produce an appropriate NPC reaction. This impacts how events

are represented in the product, in addition to guiding how priorities are assigned to different in-game events.

- The developer tool released by Bethesda for mod creation, *Skyrim Creation Kit*, gives third-party developers access to game assets and scripting tools used to create *The Elder Scrolls V: Skyrim*. Using this tool for product implementation means that its development limited to the provided game assets and scripting language, Papyrus, for implementation.

B.2.2 Naming Conventions and Definitions

- **CRPG**: Computer role-playing game
- **Event**: A game scenario that affects NPCs
- **Expressive Behaviour**: NPC actions that convey their internal emotional state
- **IN**: Input
- **Mod**: Modification, typically an extension to an existing game created by third-party developers
- **NPC**: Non-player character
- **OUT**: Output

B.2.3 Relevant Facts and Assumptions

Facts

- The *Skyrim Creation Kit* gives third-party developers access to game assets used in *The Elder Scrolls V: Skyrim* to create extensions to the base game
- The *Skyrim Creation Kit* is free for non-commercial and non-profit work
- There is a large modding community for *The Elder Scrolls V: Skyrim*, which can potentially help with issues that arise during development

Assumptions

- The *Skyrim Creation Kit* contains enough character models and animations to convey basic emotions
- The Papyrus scripting language can be used to implement all required elements of the proposed product

- Both the cognitive appraisal and psycho-evolutionary synthesis psychological models can be converted into a computerized form
- The proof-of-concept version of this product will not be commercially released
- The product can be translated into other programming languages for use in other games

B.2.4 The Scope of the Work

The Current Situation

- NPC behaviours are often created as scripts such that each behaviour is designed and implemented manually
- Richard Lazarus's cognitive appraisal and similar processes have been used previously in computer simulations of emotion
- It is unknown if Robert Plutchik's psycho-evolutionary synthesis has been implemented in a computerized system
- Similar systems for video games have been developed; most of them use the Oronty, Collins, and Clore (OCC) psychological model, but often provide little justification of their choice from a psychological standpoint

The Context of the Work

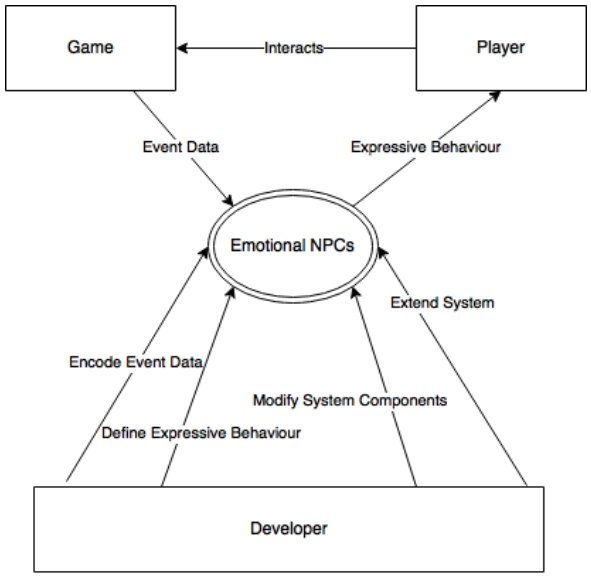


Figure B.1: External system events

Work Partitioning

Event Name	**Input and Output**	**Summary**
Event Data in the Game Environment	Event Data (IN)	Record the event data in the processing queue
Produce Expressive Behaviour	Change in animations or actions (OUT)	If the internal emotional state reaches a threshold for a defined behaviour, trigger the behaviour
Encoding Event Data	Event information (IN)	Add event information to the event database
Defining Expressive Behaviour	Animations (IN), Behaviour Script (IN)	If a developer defines a new behaviour expression, the system triggers it when the defined conditions are met
Modifying System Components	New, valid, processing rules (IN)	If a modification is made to a system component that is legal and valid, the system will be able to use the new processing rule without further modifications
Extending the System	New, valid, system module (IN)	If a system module, which is legal and valid, is added to the system in a valid extension location, the system will be able to start using the new module without further modifications

B.2.5 The Scope of the Product

Product Boundary

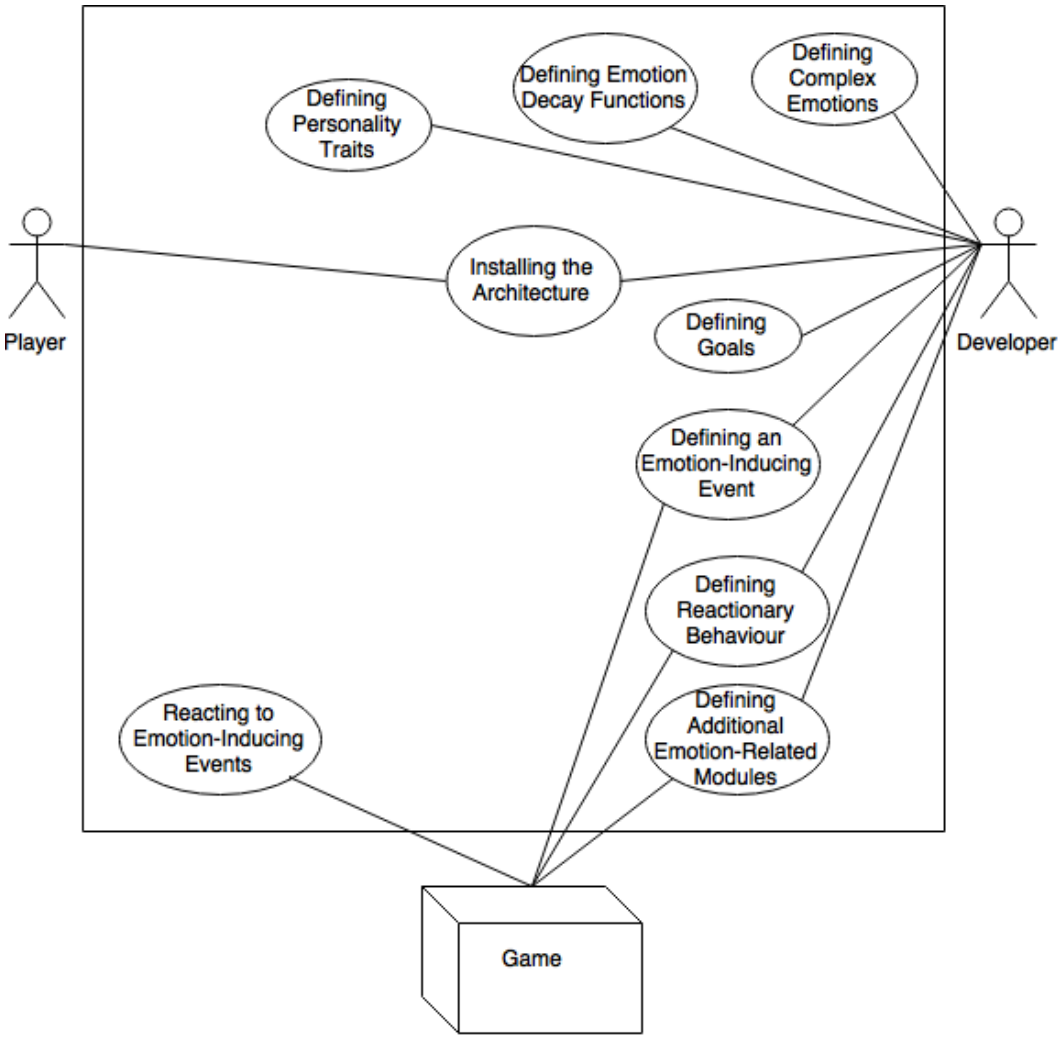


Figure B.2: Use cases across the product boundary

Individual Product Use Cases

PUC No. 1	**Reacting to emotion-inducing events**
Trigger	An event that has been identified as emotion-inducing is recognized by an NPC
Preconditions	The event is in the NPC's event database
Procedure	1. The event identifiers are captured by the NPC's emotion architecture 2. The identifiers are processed with the cognitive appraisal theory and a list of NPC goals 3. The resulting emotion values are used to update the NPC's internal emotion state 4. If the values in the emotion state reach a predefined threshold, a change in the NPC's behaviour and animations is triggered
Outcome	If the event causes a significant enough change in the NPC's internal emotional state, a change in NPC behaviour is observed; otherwise, no change is observed

PUC No. 2	**Defining an emotion-inducing event**
Trigger	An in-game event, NPC, or object should cause an NPC reaction
Preconditions	N/A
Procedure	1. Isolate the key traits of the event, NPC, or object 2. Add an identifier to the isolated traits 3. If the identifier is not defined, add information such as object categorization (e.g. "Enemy") and future consequences (e.g. "–HP")
Outcome	The new event, NPC, or object can be processed by the emotion architecture

PUC No. 3	**Defining reactionary behaviour**
Trigger	A specific emotional state should cause an NPC to behave differently
Preconditions	A set of behaviours and animations is already defined
Procedure	1. Define the range of values that characterize the emotional state 2. Add a behaviour expression unit containing the desired behaviour and animations to the emotion architecture 3. Connect the defined emotion state and behaviour unit
Outcome	The desired behaviour triggers whenever the NPC's internal emotional state is within the defined emotional state

PUC No. 4	**Defining goals**
Trigger	A new NPC state that should be maintained is defined
Preconditions	The goal can be quantified
Procedure	1. Rewrite the goal in a quantifiable form that can be understood by the system 2. Assign the goal a priority in relation to other defined NPC goals 3. Add the goal to the NPC's database
Outcome	The goal is compared to incoming event information to see if the state is maintained

PUC No. 5	**Defining complex emotions**
Trigger	NPC behaviour cannot be represented fully with basic emotions
Preconditions	The complex emotion can be represented as a combination of basic emotions, intensity values, and responsibility assignments
Procedure	1. Define the range of values that characterize the complex emotion (e.g. High intensity + Anger = Rage; Fear + Surprise + Player = Awe) 2. Add the complex emotion to the NPC's database
Outcome	A behaviour expression unit can be triggered by referring to a complex emotion definition

PUC No. 6	**Defining emotion decay functions**
Trigger	Emotions should return to equilibrium values over time
Preconditions	The decay function ensures that opposing emotions in the psycho-evolutionary synthesis model are rebalanced
Procedure	1. An equilibrium emotion state is defined 2. Define an equation that represents the emotional decay as a function of time 3. Define a trigger that runs the decay function every time it is fired 4. Connect the decay function to the emotion state
Outcome	NPC emotions are returned to the defined equilibrium values after a specified amount of time

PUC No. 7	**Defining personality traits**
Trigger	An NPC should have a unique reaction to certain in-game events, NPCs, and objects
Preconditions	The personality traits can be represented as a series of cognitive appraisal process modifiers
Procedure	1. Encode personality traits as evaluation modifiers 2. Connect the modifiers to the cognitive appraisal procedure
Outcome	The NPC's cognitive appraisal process produces different emotional values than other NPCs in response to the same in-game event, NPC, or object

PUC No. 8	**Defining additional emotion-related modules**
Trigger	Additional functionality is needed in the system
Preconditions	The module does not replace basic architecture modules, only augments them; The module is designed based on the cognitive appraisal and psycho-evolutionary synthesis theories
Procedure	1. Affected modules are chosen, including input sources and augmentation points 2. The module is attached at the identified input and augmentation points
Outcome	The system can be tested with the additional modules

PUC No. 9	**Installing the architecture**
Trigger	The architecture is not yet installed
Preconditions	*The Elder Scrolls V: Skyrim* is already installed on the target system
Procedure	1. Install the architecture as a *The Elder Scrolls V: Skyrim* mod package
Outcome	The game runs with the basic architecture and sample NPCs

B.3 Functional Requirements

ID: F–1	**PUC**: 1, 2
Type	Functional
Description	The product shall monitor and record environmental events
Rationale	The product must be able to identify when an event has occurred and create a record with event information for processing
Fit Criterion	The product maintains an event list of unprocessed events
Satisfaction: 1	**Dissatisfaction**: 5
Priority: Very High	**Conflicts**: N/A
Created:	June 14, 2016 (Edited January 20, 2017)

ID: F–2	**PUC**: 1, 4
Type	Functional
Description	The product shall maintain a prioritized list of NPC goals
Rationale	Goals are important for identifying which events are relevant to an NPC, including how they are relevant and their priority
Fit Criterion	The product is able to maintain a referable list of prioritized goals
Satisfaction: 3	**Dissatisfaction**: 1
Priority: Moderate	**Conflicts**: N/A
Created:	June 14, 2016 (Edited January 20, 2017)

ID: F–3	**PUC**: 1, 4, 8
Type	Functional
Description	The product shall compare event information with NPC goals
Rationale	The ability to recognize if an event affects an NPC's goal aids in information filtering and processing direction
Fit Criterion	The product can match event information with goal concerns
Satisfaction: 1	**Dissatisfaction**: 5
Priority: High	**Conflicts**: N/A
Created:	June 14, 2016 (Edited January 20, 2017)

ID: F–4	**PUC**: 1, 7, 8
Type	Functional
Description	The product shall convert event information into emotional values using goals
Rationale	There must be a transformation process that converts event information into emotional values; goals help by mapping events and concerns
Fit Criterion	Event information can be translated into reasonable emotional values using emotion categories from the psycho-evolutionary synthesis model
Satisfaction: 3	**Dissatisfaction**: 5
Priority: Very High	**Conflicts**: N/A
Created:	June 14, 2016 (Edited January 20, 2017)

ID: F–5	**PUC**: 1, 4
Type	Functional
Description	The product shall interrupt event processing if an unprocessed event affects a higher priority goal
Rationale	A goal with a higher priority should be processed first such that it appears that the NPC is able to reason about the current situation (e.g. being attacked should be processed before losing a game)
Fit Criterion	An incoming event that affects a high-priority goal stops any current event processing, and the incoming event is moved to the first position in the queue
Satisfaction: 5	**Dissatisfaction**: 3
Priority: Moderate	**Conflicts**: N/A
Created:	June 14, 2016 (Edited January 20, 2017)

ID: F–6	**PUC**: 1
Type	Functional
Description	The product shall maintain a list of current emotional values
Rationale	If the values from processing were used to direct NPC behaviour alone, non-immediate effects would not be possible
Fit Criterion	The product is able to maintain a list of emotion values using classifications from psycho-evolutionary synthesis
Satisfaction: 1	**Dissatisfaction**: 5
Priority: Very High	**Conflicts**: N/A
Created:	June 14, 2016 (Edited January 20, 2017)

ID: F–7	**PUC**: 1
Type	Functional
Description	The product shall update current emotion values with the values produced from event processing
Rationale	Updating the current emotion values with the values produced from events will create the effect of emotional changes
Fit Criterion	Incoming emotion values trigger a change in the stored emotion values
Satisfaction: 1	**Dissatisfaction**: 5
Priority: Very High	**Conflicts**: N/A
Created:	June 14, 2016 (Edited January 20, 2017)

ID: F–8	**PUC**: 1, 6, 8
Type	Functional
Description	The product shall be able to rebalance emotion values
Rationale	The emotion categories in psycho-evolutionary synthesis are arranged into opposing pairs, so a change in one half should trigger an equal and opposite change in the other half
Fit Criterion	A change made to one half of an emotion pair should trigger a rebalancing of emotion values to reflect an equal and opposite change in the other member of the pair
Satisfaction: 1	**Dissatisfaction**: 5
Priority: Moderate	**Conflicts**: N/A
Created:	June 14, 2016 (Edited January 20, 2017)

ID: F–9	**PUC**: 1, 6, 8
Type	Functional
Description	The product shall return stored emotion values to a base state over time
Rationale	In the absence of emotion-inducing events, it can be expected that NPCs gradually return to the state that they maintain when no events have occurred
Fit Criterion	Emotional values return to a predetermined state over a period of time that is defined by a game designer
Satisfaction: 3	**Dissatisfaction**: 4
Priority: High	**Conflicts**: N/A
Created:	June 14, 2016 (Edited January 20, 2017)

ID: F–10	**PUC**: 1, 5
Type	Functional
Description	The product shall recognize complex emotions that have been defined as combinations of psycho-evolutionary emotions and emotion intensities
Rationale	The emotions defined by psycho-evolutionary synthesis might not be enough to define the type of behaviour desired; The product must be able to recognize user-defined emotion definitions in order for them to be utilized
Fit Criterion	When a complex emotion is defined and triggered, the system should output the complex emotion instead of its individual parts
Satisfaction: 5	**Dissatisfaction**: 3
Priority: Low	**Conflicts**: N/A
Created:	June 14, 2016 (Edited January 20, 2017)

ID: F–11	**PUC**: 5
Type	Functional
Description	The product shall maintain a referable list of user-defined complex emotions
Rationale	In order for complex emotions to be used, they must be listed so that the product can refer to them
Fit Criterion	A list of user-defined emotions can be referred to and easily extended with new emotion definitions
Satisfaction: 5	**Dissatisfaction**: 3
Priority: Low	**Conflicts**: N/A
Created:	June 14, 2016 (Edited January 20, 2017)

ID: F–12	**PUC**: 1, 3, 8
Type	Functional
Description	The product shall select a class of NPC behaviours in response to changes in the emotional state
Rationale	Not all changes in emotion values result in a different behaviour; the product must be able to recognize when a significant change has occurred and what set of behaviours should be used in response
Fit Criterion	If the emotional state meets the criteria for a different behaviour class than is currently selected, the system must switch to the new behaviour class
Satisfaction: 1	**Dissatisfaction**: 5
Priority: Very High	**Conflicts**: N/A
Created:	June 14, 2016 (Edited January 20, 2017)

ID: F–13	**PUC**: 1, 3
Type	Functional
Description	The product shall interrupt current NPC actions and replace it with the actions associated with the new behaviour class
Rationale	If a change in the emotion state triggers a new behaviour class, the actions associated with that class should be used instead of the current set of actions
Fit Criterion	If a new behaviour class has been triggered, the NPC should stop its current action and begin behaving according to the actions in the new behaviour class
Satisfaction: 1	**Dissatisfaction**: 5
Priority: Very High	**Conflicts**: N/A
Created:	June 14, 2016 (Edited January 20, 2017)

ID: F–14	**PUC**: 1, 3
Type	Functional
Description	The product shall interrupt current NPC animations and replace it with the animations associated with the new behaviour class
Rationale	Body language and facial expression are a key factor in communicating human emotions, so the NPC animations should reflect the current emotional state; it might sometimes be appropriate to trigger new NPC animations, but not new NPC actions
Fit Criterion	If a new behaviour class has been triggered, the NPC should stop its current animations and start animations from the new behaviour class
Satisfaction: 1	**Dissatisfaction**: 5
Priority: Very High	**Conflicts**: N/A
Created:	June 14, 2016 (Edited January 20, 2017)

ID: F–15	**PUC**: 2
Type	Functional
Description	The product shall maintain a list of event identifiers, traits, and future consequences
Rationale	Adding tags to individual game entities is time-consuming and error-prone; a list that can be referenced is easier to maintain and extend
Fit Criterion	A list containing traits and consequences can be referenced by the processing stage and easily extended with new events
Satisfaction: 5	**Dissatisfaction**: 3
Priority: Very High	**Conflicts**: N/A
Created:	June 14, 2016 (Edited January 20, 2017)

ID: F–16	**PUC**: 7, 8
Type	Functional
Description	The product shall recognize processing modifiers and utilize them during event processing
Rationale	Personality traits that cannot be represented through behaviours must be applied to the processing of events; these modifiers must be recognized and used in order to produce varied responses to the same event
Fit Criterion	A process modifier which alters the produced emotion values (when compared to an unaltered processing unit) can be added to the product in response to the same event
Satisfaction: 3	**Dissatisfaction**: 2
Priority: Low	**Conflicts**: N/A
Created:	June 14, 2016 (Edited January 20, 2017)

ID: F–17	**PUC**: 8
Type	Functional
Description	The product shall interface with user-defined modules
Rationale	If a game developer wants to augment the functions and processing of the product, there should be minimal issues during integration
Fit Criterion	A game designer is able to find an integration location and insert their module within one hour
Satisfaction: 5	**Dissatisfaction**: 3
Priority: Very Low	**Conflicts**: N/A
Created:	June 14, 2016 (Edited January 20, 2017)

ID: F–18	**PUC**: 9
Type	Functional
Description	The product shall be installable as a "mod" for *The Elder Scrolls V: Skyrim*
Rationale	Mods are the main way that user-created content can be introduced into *The Elder Scrolls V: Skyrim*; creating the product as a mod also grants access to the game assets and scripts that make up the base game
Fit Criterion	The product can be installed in *The Elder Scrolls V: Skyrim* as a mod package
Satisfaction: 1	**Dissatisfaction**: 5
Priority: Very High	**Conflicts**: N/A
Created:	June 14, 2016 (Edited January 20, 2017)

B.4 Non-Functional Requirements

B.4.1 Look and Feel Requirements

ID: NF–1	**PUC**: 1
Type	Non-functional (Look and Feel – Style)
Description	Sample database entries, such as goals, shall adhere to the game lore of *The Elder Scrolls V: Skyrim*
Rationale	Cultural norms and expectations are defined as part of the game lore and should be considered in order to create behaviours that "make sense" for the game world
Fit Criterion	A player familiar with *The Elder Scrolls V: Skyrim* should feel that reactionary behaviours are believable within the context of the game
Satisfaction: 5	**Dissatisfaction**: 5
Priority: Very High	**Conflicts**: N/A
Created:	June 7, 2016 (Edited January 20, 2017)

ID: NF–2	**PUC**: 2 – 8
Type	Non-functional (Look and Feel – Style)
Description	The product shall appear organized
Rationale	An organized product is easier to use, which will encourage developers to give it a try
Fit Criterion	A developer that has never used the product before should feel that they can quickly find the information required to begin modifying the system
Satisfaction: 1	**Dissatisfaction**: 5
Priority: High	**Conflicts**: N/A
Created:	June 7, 2016 (Edited January 20, 2017)

B.4.2 Usability and Humanity Requirements

Ease of Use Requirements

ID: NF–3	**PUC**: 1
Type	Non-functional (Usability and Humanity – Ease of Use)
Description	The product shall cause players to be more engaged with the game
Rationale	If a player is engaged with a game that uses this system, it usually increases player satisfaction, entertainment, and willingness to continue to play the game
Fit Criterion	During a play session, if the player does not know how much time has gone by, they are sufficiently engaged
Satisfaction: 5	**Dissatisfaction**: 1
Priority: Moderate	**Conflicts**: N/A
Created:	June 7, 2016 (Edited January 20, 2017)

ID: NF–4	**PUC**: 2 – 8
Type	Non-functional (Usability and Humanity – Ease of Use)
Description	The product shall be easy to modify and extend for a game designer with an Object-Oriented programming background
Rationale	If the product cannot be easily modified to suit a particular game or extended to include more complex emotional processing components, game developers will most likely not use this system
Fit Criterion	A game developer should be able to introduce a modification or pre-developed extension to the system within a three day span
Satisfaction: 5	**Dissatisfaction**: 1
Priority: Moderate	**Conflicts**: N/A
Created:	June 7, 2016 (Edited January 20, 2017)

Learning Requirements

ID: NF–5	**PUC**: 2 – 8
Type	Non-functional (Usability and Humanity – Learning)
Description	The product shall be able to be used by game developers with little or no psychology background
Rationale	Most game designers are untrained in the field of psychology, so it is unrealistic to expect them to know the psychological theories behind the system
Fit Criterion	N/A
Satisfaction: 2	**Dissatisfaction**: 5
Priority: N/A	**Conflicts**: N/A
Created:	June 7, 2016 (Edited January 20, 2017)

ID: NF–6	**PUC**: 2 – 8
Type	Non-functional (Usability and Humanity – Learning)
Description	This product shall be used by game developers with an Object-Oriented programming background
Rationale	Many modern games are developed as Object-Oriented programs, so it is reasonable to assume that most game developers that use this system have some background in Object-Oriented design
Fit Criterion	N/A
Satisfaction: 2	**Dissatisfaction**: 5
Priority: Moderate	**Conflicts**: N/A
Created:	June 7, 2016 (Edited January 20, 2017)

ID: NF–7	**PUC**: 2 – 8
Type	Non-functional (Usability and Humanity – Learning)
Description	A game developer should be able to confidently use this product within a week's time
Rationale	Game development is a rapid and stressful process, so any tool or system that cannot be picked up easily will not be adopted by the game industry
Fit Criterion	Out of a group of 10 game developers with *The Elder Scrolls V: Skyrim* mod creation experience, eight of them should be able to use this system confidently after one week of practice
Satisfaction: 5	**Dissatisfaction**: 2
Priority: Low	**Conflicts**: N/A
Created:	June 7, 2016 (Edited January 20, 2017)

Understandability and Politeness Requirements

ID: NF–8	**PUC**: 1
Type	Non-functional (Usability and Humanity – Understandability and Politeness)
Description	The product shall adhere to common expectations regarding the connection between the provided goals, events, and NPC behaviour
Rationale	Even though they are intended as examples, the database elements that are included with the basic architecture should still adhere to commonly held beliefs about how people react to different events
Fit Criterion	With no developer modifications, the system should still aid in the believability of NPCs and overall player engagement
Satisfaction: 3	**Dissatisfaction**: 3
Priority: Moderate	**Conflicts**: N/A
Created:	June 7, 2016 (Edited January 20, 2017)
ID: NF–9	**PUC**: 1 – 8
Type	Non-functional (Usability and Humanity – Understandability and Politeness)
Description	The product shall not use psychology jargon in any of its descriptions
Rationale	Most game designers are untrained in the field of psychology, so it is unrealistic to expect them to understand terms and definitions unique to that field
Fit Criterion	A designer modifying the system should not be confused by the terminology and descriptions used
Satisfaction: 3	**Dissatisfaction**: 5
Priority: High	**Conflicts**: N/A
Created:	June 7, 2016 (Edited January 20, 2017)

B.4.3 Performance Requirements

Speed and Latency Requirements

ID: NF–10	**PUC**: 1
Type	Non-functional (Performance – Speed and Latency)
Description	The product shall respond quickly to in-game events
Rationale	If the player expects an event to elicit an emotional response from an NPC, they would also expect that it occur at the same speed as a human response
Fit Criterion	If an event should result in alternative NPC behaviour, the transition should occur in less than one second for 90% of the instances; no response should take more than 2 seconds
Satisfaction: 1	**Dissatisfaction**: 5
Priority: Very High	**Conflicts**: N/A
Created:	June 10, 2016 (Edited January 20, 2017)

ID: NF–11	**PUC**: 1
Type	Non-functional (Performance – Speed and Latency)
Description	The product shall scan the environment for new events every 0.5 seconds
Rationale	Emotion-inducing events often cause people to react quickly, which implies that they are constantly monitoring their environment for new stimuli
Fit Criterion	The system can gather information every 0.5 seconds
Satisfaction: 1	**Dissatisfaction**: 5
Priority: High	**Conflicts**: N/A
Created:	June 10, 2016 (Edited January 20, 2017)

Reliability and Availability Requirements

ID: NF–12	**PUC**: 1
Type	Non-functional (Performance – Reliability and Availability)
Description	The product shall run as long as an NPC is active
Rationale	Emotions are part of a person's ability to evaluate and navigate their environment, so it is important that an NPC's emotion architecture is able to run at all times while they are part of the in-game environment
Fit Criterion	An NPC's emotion architecture is functional as long as they are part of a loaded game level or area
Satisfaction: 3	**Dissatisfaction**: 3
Priority: Moderate	**Conflicts**: N/A
Created:	June 10, 2016 (Edited January 20, 2017)

Robustness or Fault-Tolerance Requirements

ID: NF–13	**PUC**: 1, 4
Type	Non-functional (Performance – Robustness or Fault-Tolerance)
Description	In an environment where the event load is high, the product must still be able to process high-priority events
Rationale	In an environment with an abnormally high number of emotion-inducing events, the product must always process events that affect high priority goals first, even if other event information must be discarded to handle the request
Fit Criterion	In an environment with thirty simultaneous events, the system should produce a reaction that can be traced to its highest priority goal
Satisfaction: 3	**Dissatisfaction**: 4
Priority: Moderate	**Conflicts**: N/A
Created:	June 10, 2016 (Edited January 20, 2017)

ID: NF–14	**PUC**: 1, 6
Type	Non-functional (Performance – Robustness or Fault-Tolerance)
Description	The product shall return to equilibrium values in the absence of emotion-inducing events
Rationale	In the absence of further events, humans eventually return to an "equilibrium" emotional state; the system should be able to simulate this decay process instead of immediately reverting back to equilibrium values or maintaining the new state indefinitely
Fit Criterion	In the absence of further events, an NPC should return to its original behaviour over a period of time proportional to the intensity of the current emotional state
Satisfaction: 4	**Dissatisfaction**: 3
Priority: Moderate	**Conflicts**: N/A
Created:	June 10, 2016 (Edited January 20, 2017)

Capacity Requirements

ID: NF–15	**PUC**: 1
Type	Non-functional (Performance – Capacity)
Description	The product shall be able to take in at least 20 events for processing at any given time
Rationale	A game environment can become hectic; even if not all events are of concern to an NPC's system, it is better to collect all information at first and omit some of it from further processing rather than processing them in the order that they arrived in; this also enables the system to recognize high priority information quickly
Fit Criterion	The product should be able to queue at least 20 sets of event data at any given time
Satisfaction: 1	**Dissatisfaction**: 5
Priority: Very High	**Conflicts**: N/A
Created:	June 10, 2016 (Edited January 20, 2017)

ID: NF–16	**PUC**: 1
Type	Non-functional (Performance – Capacity)
Description	The product shall be able to run in an environment with a minimum of ten identical products running simultaneously
Rationale	Most CRPGs include areas, such as towns, where there is a high concentration of NPCs; if the product requires extensive resources to run, it might become infeasible to use in games with heavily populated environments
Fit Criterion	Ten NPCs using the architecture should be able to react to an event without increasing the game's latency or suffer significant frame rate drops
Satisfaction: 1	**Dissatisfaction**: 5
Priority: Moderate	**Conflicts**: N/A
Created:	June 10, 2016 (Edited January 20, 2017)

Scalability and Extensibility Requirements

ID: NF–17	**PUC**: 2
Type	Non-functional (Performance – Scalability and Extensibility)
Description	The product should be able to handle user-defined event classifications
Rationale	The basic product might not include all event classifications (e.g. health loss) that a game developer might need to implement NPC goals, processing, and emotional values
Fit Criterion	The event classification system can be modified with an additional identification class by only modifying event processing modules
Satisfaction: 1	**Dissatisfaction**: 5
Priority: Moderate	**Conflicts**: N/A
Created:	June 10, 2016 (Edited January 20, 2017)

ID: NF–18	**PUC**: 4
Type	Non-functional (Performance – Scalability and Extensibility)
Description	The product should be capable of handling modifications to existing modules
Rationale	The basic product will only contain essential information required to provide a proof-of-concept system, which might not include all the information that a game developer needs to generate desired NPC behaviours
Fit Criterion	A game developer should be able to add a basic NPC goal to the system by only modifying the goal database
Satisfaction: 1	**Dissatisfaction**: 5
Priority: Moderate	**Conflicts**: N/A
Created:	June 10, 2016 (Edited January 20, 2017)

ID: NF–19	**PUC**: 3, 5 – 8
Type	Non-functional (Performance – Scalability and Extensibility)
Description	The product should be capable of interfacing with user-defined modules
Rationale	The basic system might not include information (e.g. social relationships) which a game developer might consider essential for their design
Fit Criterion	A game developer should be able to implement their desired module and attach it to the system without altering unrelated base modules
Satisfaction: 1	**Dissatisfaction**: 5
Priority: Moderate	**Conflicts**: N/A
Created:	June 10, 2016 (Edited January 20, 2017)

B.4.4 Operational and Environmental Requirements

Requirements for Interfacing with Adjacent Systems

ID: NF–20	**PUC**: 1 – 3
Type	Non-functional (Operational and Environmental – Interfacing with Adjacent Systems)
Description	The product shall be integrable with *The Elder Scrolls V: Skyrim*
Rationale	*The Elder Scrolls V: Skyrim* scripts are written in Papyrus, a scripting language created by Bethesda Softworks – other languages cannot be integrated into Papyrus scripts, so product implementation is constrained to the limitations of Papyrus
Fit Criterion	The product shall be runnable in *The Elder Scrolls V: Skyrim*
Satisfaction: 2	**Dissatisfaction**: 5
Priority: Very High	**Conflicts**: N/A
Created:	June 8, 2016 (Edited January 20, 2017)

Productization Requirements

ID: NF–21	**PUC**: 9
Type	Non-functional (Operational and Environmental – Productization)
Description	The product shall be installable as a "mod" package for *The Elder Scrolls V: Skyrim*
Rationale	Users and testers familiar with modding should be able to install the product without additional assistance
Fit Criterion	A user familiar with mod installation should be able to install it without asking for assistance
Satisfaction: 5	**Dissatisfaction**: 5
Priority: Very High	**Conflicts**: N/A
Created:	June 7, 2016 (Edited January 20, 2017)

Release Requirements

ID: NF–22	**PUC**: 9
Type	Non-functional (Operational and Environmental – Release)
Description	The product shall be released once as part of a graduate thesis
Rationale	Providing an initial implementation of the system with preliminary player feedback can strengthen the system's value as a proof-of-concept
Fit Criterion	A complete product can be presented to a panel of professors as part of a thesis seminar and defence process
Satisfaction: 3	**Dissatisfaction**: 3
Priority: High	**Conflicts**: N/A
Created:	June 8, 2016 (Edited January 20, 2017)

ID: NF–23	**PUC**: 9
Type	Non-functional (Operational and Environmental – Release)
Description	The product shall be released once to the *The Elder Scrolls V: Skyrim* modding community
Rationale	Even if the product is accepted in the academic community, its purpose is still to improve the experience of developers and players, who will have different expectations – releasing it to the modding community can generate feedback concerning the potential of the product as a good addition to CRPGs
Fit Criterion	The product can be uploaded to the *The Elder Scrolls V: Skyrim* modding website, Skyrim Nexus
Satisfaction: 5	**Dissatisfaction**: 2
Priority: Low	**Conflicts**: N/A
Created:	June 8, 2016 (Edited January 20, 2017)

B.4.5 Maintainability and Support Requirements

Maintenance Requirements

ID: NF–24	**PUC**: 1 – 9
Type	Non-functional (Maintainability and Support – Maintenance)
Description	The product shall be maintained by persons who are not the original designer
Rationale	If game developers are modifying existing modules, creating new ones, or creating a new game that incorporates this product, it is unrealistic to expect the original designer to maintain the core system due to the number of possible changes that could require maintenance
Fit Criterion	A game developer should be able to modify and extend the product without the guidance of the original designer
Satisfaction: 3	**Dissatisfaction**: 5
Priority: Moderate	**Conflicts**: N/A
Created:	June 9, 2016 (Edited January 20, 2017)

Supportability Requirements

ID: NF–25	**PUC**: 2 – 8
Type	Non-functional (Maintainability and Support – Supportability)
Description	The product shall have essential usage information included in Papyrus scripts
Rationale	Most game developers will only need to add new code to existing modules in the product, so essential information, such as adding a new NPC goal, should be included as part of the in-script documentation
Fit Criterion	A game developer who has not used the system before should not need external product documentation in order to modify existing product modules
Satisfaction: 5	**Dissatisfaction**: 3
Priority: Moderate	**Conflicts**: N/A
Created:	June 9, 2016 (Edited January 20, 2017)

ID: NF–26	**PUC**: 2 – 8
Type	Non-functional (Maintainability and Support – Supportability)
Description	The product shall have a companion usage manual
Rationale	If a game developer wants to learn about system details (e.g. how each module works) or make additional product modules, additional documentation should be provided to help them decide how their modules should be designed and where to connect them
Fit Criterion	A game designer should be able to create and connect a simple module to the existing product
Satisfaction: 5	**Dissatisfaction**: 2
Priority: Low	**Conflicts**: N/A
Created:	June 9, 2016 (Edited January 20, 2017)

Adaptability Requirements

ID: NF–27	**PUC**: 1 – 9
Type	Non-functional (Maintainability and Support – Adaptability)
Description	The product will be implemented using Papyrus, but it might be ported to other programming or scripting languages in the future
Rationale	The scope of this project only requires a proof-of concept system, and the Papyrus scripting language cannot interface with other programming or scripting languages; it is possible that this product could be ported to other languages, such as UDK, for use in other games
Fit Criterion	The product is written in a way that can be understood independently of the Papyrus scripting language
Satisfaction: 5	**Dissatisfaction**: 1
Priority: Low	**Conflicts**: N/A
Created:	June 9, 2016 (Edited January 20, 2017)

B.4.6 Cultural and Political Requirements

ID: NF–28	**PUC**: 9
Type	Non-functional (Cultural)
Description	The product shall adhere to standard Object-Oriented design practices
Rationale	The scripting language supported by the Creation Kit, Papyrus, is based on Object-Oriented design principles
Fit Criterion	A programmer versed in Object-Oriented design should be able to recreate the architecture as a set of classes and objects
Satisfaction: 5	**Dissatisfaction**: 5
Priority: Very High	**Conflicts**: N/A
Created:	June 7, 2016 (Edited January 20, 2017)

B.4.7 Legal Requirements

Compliance Requirements

ID: NF–29	**PUC**: N/A
Type	Non-functional (Legal – Compliance)
Description	The product shall adhere to all terms and conditions of Bethesda's Creation Kit End User License Agreement (EULA)
Rationale	The Creation Kit EULA must be followed
Fit Criterion	The terms set out in the Creation Kit EULA are followed
Satisfaction: 1	**Dissatisfaction**: 5
Priority: Very High	**Conflicts**: N/A
Created:	June 7, 2016 (Edited January 20, 2017)

ID: NF–30	**PUC**: N/A
Type	Non-functional (Legal – Compliance)
Description	The product shall adhere to all terms and conditions of McMaster's Academic Integrity policy
Rationale	McMaster's Academic Integrity policy must be followed
Fit Criterion	McMaster's Academic Integrity policy is followed
Satisfaction: 1	**Dissatisfaction**: 5
Priority: Very High	**Conflicts**: N/A
Created:	June 7, 2016 (Edited January 20, 2017)

Standards Requirements

ID: NF–31	**PUC**: N/A
Type	Non-functional (Legal – Standards)
Description	The product shall adhere to all standards set out by Dr. Carette
Rationale	As the thesis supervisor, Dr. Carette has expected standards of the architecture and its development
Fit Criterion	Dr. Carette approves the design and final product
Satisfaction: 1	**Dissatisfaction**: 5
Priority: Very High	**Conflicts**: N/A
Created:	June 7, 2016 (Edited January 20, 2017)

B.5 Project Issues

B.5.1 Open Issues

It is unclear at this point if:

- The Papyrus scripting language and game assets in the Creation Kit will be enough to implement the proposed functionality

- The psychological models chosen will produce a noticeable effect

B.5.2 Off-the-Shelf Solutions

Ready-made Products

- GAMYGDALA
  An architecture designed to create believable emotional behaviour in video game NPCs. It bases its design on the Oronty, Clore, and Collins (OCC) model of emotion. This product differs from GAMYGDALA by implementing different models of emotion to see if equal or better effects can be achieved.

- Tok Architecture
  An architecture designed to allow content creators complete creative control over video game NPCs as narrative characters. This product differs from the Tok architecture by taking some creative control away from designers in exchange for more automation.

Reusable Components

As part of the Skyrim Creation Kit, developers have access to all models and animations that are included in *The Elder Scrolls V: Skyrim*. This reduces the amount of development time required to create this product because pre-existing models can be used to create a closed level for testing purposes. This is also essential for implementing psychological models of emotion because an important aspect of emotional displays is body language and facial expressions that accompany varied behaviours. These elements of emotional expression can be simulated with pre-existing character animations.

Products that can be Copied

- Closed levels already exist in *The Elder Scrolls V: Skyrim*; if an environment exists with at least five NPCs, it might be more prudent to remove the ability to leave the level and use it as the testing environment, which would also help highlight which events to encode and what type of responses to focus on

- Specific processing elements can be taken from GAMYGDALA and the Tok architecture to address design challenges that they have done well

B.5.3 New Problems

Effects on the Current Environment

This product will create additional processing strain in a real-time, open world game environment. This makes it increasingly important to meet identified performance requirements so the effects of extra processing do not affect other game functions (e.g. registering player inputs).

Effects on the Installed Systems

Since this product affects the behaviour of NPCs, care needs to be taken to ensure that this product does not interfere with essential NPC tasks (e.g. distribution and advancement of quests).

Potential User Problems

Part of the purpose for creating this product is to see if it enhances player engagement and entertainment. If this product should fail in this endeavour, it will negatively affect the player's perception of the game and should not be used in further projects. Another potential player problem is that they do not want this type of system in *The Elder Scrolls* games because it affects their perception of the game, and so the product needs to be tested in a different environment.

With regards to game designers, the product might not be as user-friendly and understandable as intended. Therefore, even if the product is successful with players, it might not be adapted and used in future game projects.

Limitations in the Anticipated Implementation Environment that may Inhibit the New Product

At this time, it is unclear if the Skyrim Creation Kit and Papyrus scripting language contains all of the necessary models, animations, and programming tools required to create this type of system. In the event that one or more product features cannot be included due to this limitation, the desired functionality will be re-evaluated and, if it is deemed mandatory, an alternative implementation method or approach will be attempted.

Follow-up Problems

In the event that the product does not, or negatively, affects the player's engagement and entertainment, an analysis of the system should be compiled that explores the potential reasons for this failure and provides suggestions for future attempts for creating a similar system.

B.5.4 Risks

- **The psychological models chosen do not produce noticeable or desirable effects**
  If this occurs before testing, the testing stage can be forgone and replaced with an analysis of why the chosen models did not work as expected. If the fault lies in the implementation and not the models themselves, this will need to be addressed as well. If this problem is noticed after testing, then the original plan can be followed, although with negative results.

- **The proposed architecture cannot be implemented in the Skyrim Creation Kit**
  This can occur if either the adequate number and type of assets are unavailable, or the Papyrus scripting language does not contain all the required functionality. In this case, depending on how much development time is available, two alternatives can be considered. The first is to implement the product as part of a text-based game with simple graphics, which can still test the system's ability to create believable behaviours. The downside is that there is less likely to be noticeable player engagement effects. The second alternative is more intensive, and requires that a simple level be created in the Unity game engine with assets from the Creative Commons. This option allows for more control over the testing environment, but requires significantly more development time.

- **Underestimation of the required development time**
  With no previous experience using the Skyrim Creation Kit, it is possible to underestimate the amount of time required to implement the product. In this case, product features that are not required to test player engagement and enjoyment will be moved to future development goals.

B.5.5 Costs

In order to attract enough participants to test the system within a small time frame, it might be useful to offer compensation (typically $10). However, this might not be necessary if enough interest is generated in the product.

B.5.6 User Documentation Requirements

The user documentation for this product requires:

- A layman explanation of the psychological models implemented

- An overview of the purpose for each product module and how to extend them

- Suggestions for new product modules

B.5.7 Ideas for Solutions

A well-known environment in *The Elder Scrolls V: Skyrim* is a tavern. Due to its social nature, this might be a good environment to test the architecture in while limiting the number of events and goals to encode.

Appendix C

Documentation for User Study Participants

The questionnaires and consent form given to study participants have been included to assist in the recreation of this or similar studies.

C.1 Participant Questionnaires

These questions, presented as a Google Form with an associated Google Sheet, were asked of the participants. They were designed to collect demographic and subjective information.

C.1.1 The Elder Scrolls V: Skyrim (Pre-Play Questionnaire)

1. (Text field) What is your age?
2. (Radio button) What is your gender?
    - Male
    - Female
3. (Date field) Approximately, when was the last time that you played Skyrim?
4. (Radio button) Approximately how many hours have you played Skyrim?
    - 0 – 5 hours
    - 5 – 10 hours
    - 10 – 20 hours

- 20 – 50 hours
- 50 – 100 hours
- 100+ hours

5. (Radio button) In the save file that you will use for this study, have you completed the "Civil War" quest line?
   - Yes
   - No
6. (Radio button, Optional) If you answered "Yes", which side did you complete it on?
   - Imperial
   - Stormcloak
7. (Radio button) What do you feel is Skyrim's best quality?
   - Story (includes the world lore)
   - Battle mechanics
   - NPC interactions
   - Game environment
   - (Text field) Other

C.1.2 The Elder Scrolls V: Skyrim (Post-Play Questionnaire)

1. (Radio button) Did you notice any difference between the two versions?
   - Yes
   - No
2. (Radio button, Optional) If you answered "Yes", which version of Skyrim did you prefer?
   - A
   - B
   - No preference
3. (Radio Button, Optional) Which version was the modified one?
   - A

- B

4. (Long answer field, Optional) What were those differences?
5. (Long answer field, Optional) Is there anything else that you would like to say about the two Skyrim versions?

C.2 Consent Form

This form was presented to participants before beginning the study. It informed the user of the study's purpose, what their instructions would be, and information to guide their decision to grant consent such as potential risks and benefits.

DATE: February 23, 2017

APPENDIX A
LETTER OF INFORMATION / CONSENT

A Study Improving the Believability of Non-Player Characters in Video Games

Principal Investigator:
Geneva Smith
Department of Computing and Software
McMaster University
Hamilton, Ontario, Canada
E-mail: smithgm@mcmaster.ca

Co-Investigator:
Dr. Jacques Carette
Department of Computing and Software
McMaster University
Hamilton, Ontario, Canada
E-mail: carette@mcmaster.ca

Purpose of the Study:
You are invited to take part in this study on improving the believability of non-player characters (NPCs) in role-playing video games. For the purpose of this study, an extension (mod) was created for Bethesda's *The Elder Scrolls V: Skyrim*. I would like to determine if the design created for this study is noticeable by players and, if so, if they perceive the differences as an improvement or asset to the original game. This study is being conducted as part of the thesis requirement for the Masters of Applied Science (M.A.Sc.) of Software Engineering.

Procedures involved in the Research:
You will first be given a survey that gathers basic demographics information and gauges your overall playing habits regarding *The Elder Scrolls V: Skyrim*. You will then be asked to log into your personal Steam account and download your *The Elder Scrolls V: Skyrim* save file, which is not dependant on any downloadable content (DLC) or mods.
There will be two 20 minute play sessions – one with the study mod installed and one without. You will not be told which one you are playing at any given time. Due to this factor, I will ask you to turn away before each study so that I can load the game for you, ensuring that you are unaware of which version is running. You will be able to choose your preferred game controller (keyboard and mouse or Xbox 360 controller), which will be used for both sessions. For the duration of the study, please do not leave the in-game city of Windhelm.
After you have completed both sessions, I will ask you to logout of your personal Steam account and complete a post-study questionnaire. Your participation in the study will conclude when you complete the questionnaire.

Potential Harms, Risks or Discomforts:
The risks involved in participating in this study are minimal. There are known risks with "modding" whereby some game save files will no longer work. To circumvent this, I will ask you not to save the game at any point during the study.
You do not need to answer questions that you do not want to answer or that make you feel uncomfortable. I describe below the steps I am taking to protect your privacy.

Potential Benefits
The research will not benefit you directly. We hope to learn more about different techniques to improve the believability of NPCs in video games. I hope that what is learned as a result of this

study will help us to better understand what type of techniques do or do not have the potential to increase the enjoyment of video games.

Confidentiality
You are participating in this study confidentially. I will not use your name or any information that would allow you to be identified. No one but me will know whether you were in the study unless you choose to tell them.
Once the study is complete, an archive of the data, without identifying information, will be maintained as part of the completed thesis.

Participation and Withdrawal:
Your participation in this study is voluntary and anonymous. If you decide to be part of the study, you can stop (withdraw), from the user study for whatever reason, even after signing the consent form, or part-way through the study. If you decide to withdraw, there will be no consequences to you. In cases of withdrawal, any data you have provided will be destroyed unless you indicate otherwise. If you do not want to answer some of the questions you do not have to, but you can still be in the study.
However, once you have submitted your responses for this anonymous study, your answers will be put into a database and will not be identifiable to you. This means that once you have submitted your survey, your responses cannot be withdrawn from the study because we will not be able to identify which responses are yours.

Information about the Study Results:
I expect to have this study completed by approximately April 2017. If you would like a brief summary of the results, please let me know how you would like it sent to you.

Questions about the Study:
If you have questions or need more information about the study itself, please contact me at:
smithgm@mcmaster.ca
This study has been reviewed by the McMaster University Research Ethics Board and received ethics clearance. If you have concerns or questions about your rights as a participant or about the way the study is conducted, please contact:
McMaster Research Ethics Secretariat
Telephone: (905) 525-9140 ext. 23142
C/o Research Office for Administrative Development and Support
E-mail: ethicsoffice@mcmaster.ca

CONSENT

- I have read the information presented in the information letter about a study being conducted by Geneva Smith and Dr. Jacques Carette of McMaster University.
- I have had the opportunity to ask questions about my involvement in this study and to receive additional details I requested.
- I understand that if I agree to participate in this study, I may withdraw from the study at any time or up until approximately April 2017.
- I have been given a copy of this form.
- I agree to participate in the study.

Signature: _____ Date: _____

Name of Participant (Printed) _____

1. …Yes, I would like to receive a summary of the study's results.
Please send them to me at this email address _____
Or to this mailing address: _____
_____
_____

… No, I do not want to receive a summary of the study's results.

2. I agree to use my own *The Elder Scrolls V: Skyrim* save file, knowing the possible data loss risk associated with "modding". I acknowledge that this requires the temporary use of my Steam account.
... Yes, I agree to use my own save file
... No, I do not agree to use my own save file

Bibliography

Adams, E. (2010). *Fundamentals of Game Design*. New Riders, 2nd edition.

Andrade, G., Ramalho, G., Santana, H., and Corruble, V. (2005). Extending Reinforcement Learning to Provide Dynamic Game Balancing. In *Proceedings of the IJCAI Workshop on Reasoning, Representation, and Learning in Computer Games*, pages 7–12.

Aponte, M., Levieux, G., and Natkin, S. (2009). Scaling the level of difficulty in single player video games. In *Entertainment Computing–ICEC 2009*, pages 24–35. Springer.

Bakkes, S., Spronck, P., and Postma, E. (2005). Best-Response Learning of Team Behaviour in Quake III. In *Proceedings of the IJCAI Workshop on Reasoning, Representation, and Learning in Computer Games*, pages 13–18.

Bakkes, S., Spronck, P., and den Herik, J. V. (2009). Rapid and reliable adaptation of video game AI. *Computational Intelligence and AI in Games, IEEE Transactions on*, **1**(2), 93–104.

Bock, T. (2009). Artificial emotions 2: An implementation of OCC, including middle-term dynamics. *EMBOTS*.

Bostan, B. and Öğüt, S. (2009). Game challenges and difficulty levels: Lessons learned From RPGs. In *International Simulation and Gaming Association Conference*.

Campos, J. J., Frankel, C. B., and Camras, L. (2004). On the nature of emotion regulation. *Child development*, **75**(2), 377–394.

Cannon, W. (1929). Bodily changes in pain, hunger, fear and rage. *American Journal of Psychiatry*, **86**(4).

Carlson, J. G. and Hatfield, E. (1992). *Psychology of Emotions*. Holt, Rinehart and Winston, Inc.

Carreker, D. (2012). *The Game Developer's Dictionary: A Multidisciplinary Lexicon for Professionals and Students*. Course Technology Press, 1st edition.

Chauvin, S., Levieux, G., Donnart, J.-Y., and Natkin, S. (2015). Making sense of emergent narratives: An architecture supporting player-triggered narrative processes. In *Computational Intelligence and Games*.

Clore, G. L. and Ortony, A. (2008). *Appraisal theories: How cognition shapes affect into emotion*, pages 628 – 642. Guilford Press, 3rd edition.

Colby, K. M. (1981). Modeling a paranoid mind. *Behavioral and Brain Sciences*, **4**(04), 515–534.

Cole, P. M., Hall, S. E., and Hajal, N. (2008). Emotion dysregulation as a risk factor for psychopathology. *Child and adolescent psychopathology*, **2**, 341–373.

Darwin, C. (1872). The expression of the emotions in man and animals.

de Jong, S., Spronck, P., and Roos, N. (2005). Requirements for Resource Management Game AI. In *Proceedings of the IJCAI Workshop on Reasoning, Representation, and Learning in Computer Games*, pages 43–48.

De Raad, B. (2000). *The Big Five Personality Factors: The psycholexical approach to personality.* Hogrefe & Huber Publishers.

Ekman, P. (1992). An argument for basic emotions. *Cognition & emotion*, **6**(3-4), 169–200.

Ekman, P. (1999). *Basic Emotions*, pages 45–60. John Wiley & Sons, Ltd.

Ekman, P. and O'Sullivan, M. (1991). Who can catch a liar? *American psychologist*, **46**(9), 913–920.

Elliott, C. D. (1992). The affective reasoner: A process model of emotions in a multi-agent system. *Doctoral Dissertation*.

Ellsworth, P. C. and Scherer, K. R. (2003). Appraisal processes in emotion. *Handbook of affective sciences*, pages 572–595.

Fontaine, J. R., Scherer, K. R., Roesch, E. B., and Ellsworth, P. C. (2007). The world of emotions is not two-dimensional. *Psychological science*, **18**(12), 1050–1057.

Fox, N. A., Hane, A. A., and Pine, D. S. (2007). Plasticity for affective neurocircuitry: How the environment affects gene expression. *Current Directions in Psychological Science*, **16**(1), 1–5.

Fredrickson, B. L., Cohn, M. A., Coffey, K. A., Pek, J., and Finkel, S. M. (2008). Open hearts build lives: Positive emotions, induced through loving-kindness meditation, build consequential personal resources. *Journal of Personality and Social Psychology*, **95**(5), 1045–1062.

Freud, S. (1965). *New introductory lectures on psychoanalysis (trans.)*. WW Norton.

Frijda, N. H. (1988). The laws of emotion. *American psychologist*, **43**(5), 349–358.

Frijda, N. H. and Swagerman, J. (1987). Can computers feel? Theory and design of an emotional system. *Cognition and emotion*, **1**(3), 235–257.

Gal, Y., Grosz, B., Kraus, S., Pfeffer, A., and Shieber, S. (2005). Colored Trails: A Formalism for Investigating Decision-Making in Strategic Environments. In *Proceedings of the IJCAI Workshop on Reasoning, Representation, and Learning in Computer Games*, pages 25–30.

Gebhard, P. (2005). ALMA: A layered model of affect. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 29–36. ACM.

Goldsmith, H. H., Pollak, S. D., and Davidson, R. J. (2008). Developmental neuroscience perspectives on emotion regulation. *Child Development Perspectives*, **2**(3), 132–140.

Graft, K. (2015). When artificial intelligence in games becomes...artificially intelligent. *Gamasutra*.

Griffiths, P. E. (2003). Basic Emotions, Complex Emotions, Machiavellian Emotions. *Royal Institute of Philosophy Supplement*, **52**, 39–67.

Hoffman, M. L. (2008). Empathy and prosocial behavior. *Handbook of emotions*, **3**, 440–455.

Hudlicka, E. (2002). This time with feeling: Integrated model of trait and state effects on cognition and behavior. *Applied Artificial Intelligence*, **16**(7-8), 611–641.

Hunicke, R. and Chapman, V. (2004). AI for dynamic difficulty adjustment in games. In *Challenges in Game Artificial Intelligence AAAI Workshop*, volume 2.

Izard, C. E. (2009). Emotion theory and research: Highlights, unanswered questions, and emerging issues. *Annual Review of Psychology*, **60**, 1–25.

Izard, C. E. (2010). The many meanings/aspects of emotion: Definitions, functions, activation, and regulation. *Emotion Review*, **2**(4), 363–370.

Jacobs, S., Ferrein, A., and Lakemeyer, G. (2005). Unreal GOLOG Bots. In *Proceedings of the IJCAI Workshop on Reasoning, Representation, and Learning in Computer Games*, pages 31–36.

Jung, C. G. (1966). *Two essays on analytical psychology*, volume 7. Routledge & Kegan Paul, Ltd., 2nd edition.

Kagan, J. (2007). *What is emotion? History, measures, and meanings*. Yale University Press.

Keltner, D., Oatley, K., and Jenkins, J. M. (2014). *Understanding Emotions*. John Wiley & Sons, Inc., 3rd edition.

Kondeti, B., Nallacharu, M., Youngblood, M., and Holder, L. (2005). Interfacing the D'Artagnan Cognitive Architecture to the Urban Terror First-Person Shooter Game. In *Proceedings of the IJCAI Workshop on Reasoning, Representation, and Learning in Computer Games*, pages 55–60.

Kremers, R. (2009). *Level Design: Concept, Theory, and Practice*. A K Peters, Ltd.

Lange, C. G. and James, W. (1922). *The emotions*, volume 1. Williams & Wilkins.

Lazarus, R. S. (1991). *Emotion and adaptation*. Oxford University Press.

Lazarus, R. S. (2006). *Stress and emotion: A new synthesis*. Springer Publishing Company.

Lazarus, R. S. and Lazarus, B. N. (1996). *Passion and reason: Making sense of our emotions*. Oxford University Press.

Lazarus, R. S., Kanner, A. D., Folkman, S., *et al.* (1980). Emotions: A cognitive-phenomenological analysis. *Theories of emotion*, **1**, 189–217.

LeDoux, J. (2008). Emotional coloration of consciousness: How feelings come about. In L. Weiskrantz and M. Davies, editors, *Frontiers of Consciousness*. Oxford University Press.

Lehnert, W. G., Dyer, M. G., Johnson, P. N., Yang, C., and Harley, S. (1983). BORIS - an experiment in in-depth understanding of narratives. *Artificial intelligence*, **20**(1), 15–62.

Levenson, R. W. (2003). Blood, sweat, and fears. *Annals of the New York Academy of Sciences*, **1000**(1), 348–366.

Lewis, M. D. (2005). Bridging emotion theory and neurobiology through dynamic systems modeling. *Behavioral and brain sciences*, **28**(02), 169–194.

Lucat, B. and Haahr, M. (2015). What makes a successful emergent narrative: The case of Crusader Kings II. In *Interactive Storytelling*, pages 259–266. Springer International Publishing.

Maclin, R., Shavlik, J., Walker, T., and Torrey, L. (2005). Knowledge-based Support-Vector Regression for Reinforcement Learning. In *Proceedings of the IJCAI Workshop on Reasoning, Representation, and Learning in Computer Games*, pages 61–66.

Marsella, S. C. and Gratch, J. (2009). EMA: A process model of appraisal dynamics. *Cognitive Systems Research*, **10**(1), 70–90.

Martens, C., Ferreira, J. F., Bosser, A.-G., and Cavazza, M. (2014). Generative story worlds as linear logic programs. In *Seventh Intelligent Narrative Technologies Workshop*.

Marthi, B., Russel, S., and Latham, D. (2005). Writing Stratagus-playing Agents in Concurrent ALisp. In *Proceedings of the IJCAI Workshop on Reasoning, Representation, and Learning in Computer Games*, pages 67–71.

Mayer, J. D., Roberts, R. D., and Barsade, S. G. (2008). Human abilities: Emotional intelligence. *Annual Review of Psychology*, **59**, 507–536.

Mehrabian, A. (1996). Pleasure-Arousal-Dominance: A general framework for describing and measuring individual differences in temperament. *Current Psychology*, **14**(4), 261–292.

Millington, I. and Funge, J. (2009). *Artificial Intelligence for Games*. CRC Press, 2nd edition.

Molineaux, M., Aha, D., and Ponsen, M. (2005). Defeating Novel Opponents in a Real-Time Strategy Game. In *Proceedings of the IJCAI Workshop on Reasoning, Representation, and Learning in Computer Games*, pages 72–77.

Oatley, K. and Johnson-Laird, P. N. (1987). Towards a cognitive theory of emotions. *Cognition and emotion*, **1**(1), 29–50.

Ortony, A., Clore, G. L., and Collins, A. (1988). The cognitive structure of emotions.

Plutchik, R. (1980). *Emotion: A psychoevolutionary synthesis*. Harpercollins College Division.

Plutchik, R. (2001). The nature of emotions. *American Scientist*, **89**(4), 344–350.

Ponsen, M. (2004). Improving adaptive game AI with evolutionary learning. *Thesis (Masters of Science)*.

Ponsen, M., Munoz-Avila, H., Spronck, P., and Aha, D. (2006). Automatically generating game tactics through evolutionary learning. *AI Magazine*, **27**(3), 75.

Ponsen, M., Spronck, P., Munoz-Avila, H., and Aha, D. (2007). Knowledge acquisition for adaptive game AI. *Science of Computer Programming*, **67**(1), 59–75.

Popescu, A., Broekens, J., and van Someren, M. (2014). GAMYGDALA: An emotion engine for games. *IEEE Transactions on Affective Computing*, **5**(1), 32–44.

Reilly, W. S. (1996). Believable social and emotional agents. *Doctoral Dissertation*.

Riedl, M. (2005). Towards Integrating AI Story Controllers and Game Engines: Reconciling World State Representations. In *Proceedings of the IJCAI Workshop on Reasoning, Representation, and Learning in Computer Games*, pages 84–89.

Roseman, I. J. and Elliot, A. (2008). Structure of emotions motivations and emotivations: Approach, avoidance, and other tendencies in motivated and emotional behavior. *Handbook of approach and avoidance motivation*, pages 343–366.

Rothbart, M. K. (2011). *Becoming who we are: Temperament and personality in development*. Guilford Press.

Russell, J. A. (2003). Core affect and the psychological construction of emotion. *Psychological review*, **110**(1), 145–172.

Sanchez-Pelegrin, R. and Diaz-Agudo, B. (2005). An Intelligent Decision Module based on CBR for C-evo. In *Proceedings of the IJCAI Workshop on Reasoning, Representation, and Learning in Computer Games*, pages 90–94.

Schachter, S. and Singer, J. (1962). Cognitive, social, and physiological determinants of emotional state. *Psychological review*, **69**(5), 379.

Schadd, F., Bakkes, S., and Spronck, P. (2007). Opponent modeling in real-time strategy games. In *Proceedings of the 8th International Conference on Intelligent Games and Simulation (GAME-ON 2007)*, pages 61–70.

Scherer, K. R. (2005). What are emotions? And how can they be measured? *Social science information*, **44**(4), 695–729.

Shweder, R. A., Haidt, J., Horton, R., and Joseph, C. (2008). *The Cultural Psychology of the Emotions: Ancient and Renewed*, pages 343 – 366. Guilford Press, 3rd edition.

Skinner, B. F. (1965). *Science and human behavior*. Simon and Schuster.

Sloman, A., Chrisley, R., and Scheutz, M. (2005). The architectural basis of affective states and processes. *Who needs emotions*, **32**.

Solomon, R. L. (1980). The opponent-process theory of acquired motivation: The costs of pleasure and the benefits of pain. *American psychologist*, **35**(8), 691–712.

Spronck, P. (2005). A Model for Reliable Adaptive Game Intelligence. In *Proceedings of the IJCAI Workshop on Reasoning, Representation, and Learning in Computer Games*, pages 95–100.

Spronck, P., Sprinkhuizen-Kuyper, I., and Postma, E. (2003). Online adaptation of game opponent AI in simulation and in practice. In *Proceedings of the 4th International Conference on Intelligent Games and Simulation (GAME-ON 2003)*, pages 93–100.

Spronck, P., Sprinkhuizen-Kuyper, I., and Postma, E. (2004a). Difficulty Scaling of Game AI. In *Proceedings of the 5th International Conference on Intelligent Games and Simulation (GAME-ON 2004)*, pages 33–37.

Spronck, P., Sprinkhuizen-Kuyper, I., and Postma, E. (2004b). Enhancing the performance of dynamic scripting in computer games. In *Entertainment Computing–ICEC 2004*, pages 296–307. Springer.

Thiboust, J. (2013). Focusing Creativity: RPG Genres. *Gamasutra*.

Watson, J. B. (1919). *Psychology: From the standpoint of a behaviorist*. Lippincott.

Wray, R., van Lent, M., Beard, J., and Brobst, P. (2005). The Design Space of Control Options for AIs in Computer Games. In *Proceedings of the IJCAI Workshop on Reasoning, Representation, and Learning in Computer Games*, pages 113–118.

Yannakakis, G. and Hallam, J. (2005). A Scheme for Creating Digital Entertainment with Substance. In *Proceedings of the IJCAI Workshop on Reasoning, Representation, and Learning in Computer Games*, pages 119–124.

Yannakakis, G. and Togelius, J. (2014). A Panorama of Artificial and Computational Intelligence in Games. *IEEE Xplore*.