A Chatbot Service for use in Video Game Development

Alec John Larsen

A dissertation submitted to the Faculty of Engineering and the Built Environment, University of the Witwatersrand, Johannesburg, in fulfilment of the requirements for the degree of Master of Science in Engineering.

Johannesburg, August 2014

Declaration

I declare that this dissertation is my own, unaided work, except where otherwise acknowledged. It is being submitted for the degree of Master of Science in Engineering in the University of the Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination in any other university.

Signed this ____ day of _____ 20__

_____

Alec John Larsen

Abstract

Character dialogue writing for modern digital games is a difficult process as many of them are non-linear and as such the authors have to write many different versions of the characters' dialogues to compensate for different orders that the players may perform the tasks. The proposed solution is to replace game cut-scenes with interactive dialogue using a chatbot - a computer program that simulates conversation by responding to user's text inputs in a natural language such as English. This falls into the field of interactive fiction. By investigating existing interaction fiction systems it was determined that they are difficult to author, use predefined dialogue and do not handle mood. The proposed solution allows authors to create different non-playable characters (NPCs) using a single chatbot, based on the Artificial Linguistic Internet Computer Entity (A.L.I.C.E.) open source project. This single chatbot acts as the knowledge-base for all the non-playable characters (NPCs) and provides stock responses to the player's inputs. These stock responses are then translated to match the NPC being interacted with. The translation takes the NPCs' dialects and moods into account, generating emergent dialogue. This approach simplifies the authoring approach as the knowledge-base is created once, independently of the NPCs and simple rules are defined that allow each NPC to convey the relevent persona. The general chatbot and NPC translation rules are created using a GUI. The hedonic quality of the GUI was tested by five people via a questionnaire. This showed that while the GUI is easy to use, the general chatbot workflow could be simplified. A test game was created and 35 people rated the system via a questionnaire. This gave a pleasing qualitative result as the NPCs are effective at conveying information, their responses are variable, their personas are evident through their responses and players noticed a change in mood. Theoretical analysis shows that the simple rules can produce extremely variable outputs.

For my dad. You were the best father that a son could wish for. You have shown me how to fight and to never give up. I love you. May you rest in peace.

Acknowledgements

I would like to thank my supervisors, Stephen Levitt and Hanli Geyser, for their dedication and advice. I would also like to thank my friends for their encouragement through the tribulations of the year and for keeping me sane.

I would like to thank my parents for their constant support: financial and otherwise.

The financial assistance of the National Research Foundation (NRF) towards this research is hereby acknowledged. Opinions expressed and conclusions arrived at, are those of the author and are not necessarily to be attributed to the NRF.

Contents

List of Figures

List of Tables

Nomenclature

ABL	A Behaviour Language
AI	Artificial Intelligence
AIML	Artificial Intelligence Markup Language
A.L.I.C.E	Artificial Linguistic Internet Computer Entity
ANEW	Affective Norms for English Words
CML	Character Modelling Language
DM	Drama Manager
DTML	Dialect Transforms Modelling Language
FML	Feelings Modelling Language
FPS	First Person Shooter
GUI	Graphical User Interface
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IDS	Interactive Digital Storytelling
IF	Interactive Fiction
LIWC	Linguistic Inquiry and Word Count
LTML	Lookup Template Modelling Language
LSA	Latent Semantic Analysis
NPC	Non-Playable Character
OMCS	Open Mind Common Sense

PC	Personal Computer
RAM	Random Access Memory
RNF	Response Not Found
RPG	Role Playing Game
SRE	Stimulus-Response Element
SRP	Stimulus-Response Pairs
US(A)	United States (of America)
XML	eXtensible Markup Language

Chapter 1

Introduction

The game development industry has evolved drastically and so has its audience. Originally only the developers who made the games had the technical skills and equipment needed to play digital games [1]. Digital games are now enjoyed by a much wider audience, and can now be played on arcade cabinets, consoles, PCs and mobile devices (tablets and smartphones). Digital game design requires an understanding of your target platform and audience [2]. The game development industry is a multi-billion dollar industry, and is on par with the other sections of the entertainment industry (such as music, film and books) [3]. The lucrative nature of the game development industry has lead to an increased interest in game-related research and is part of the motivation of this research. A summary of the revenue of the game development industry is presented in Figure 1.1 (the figures are based mostly on the US revenue in [3], as the global figures are not readily available).

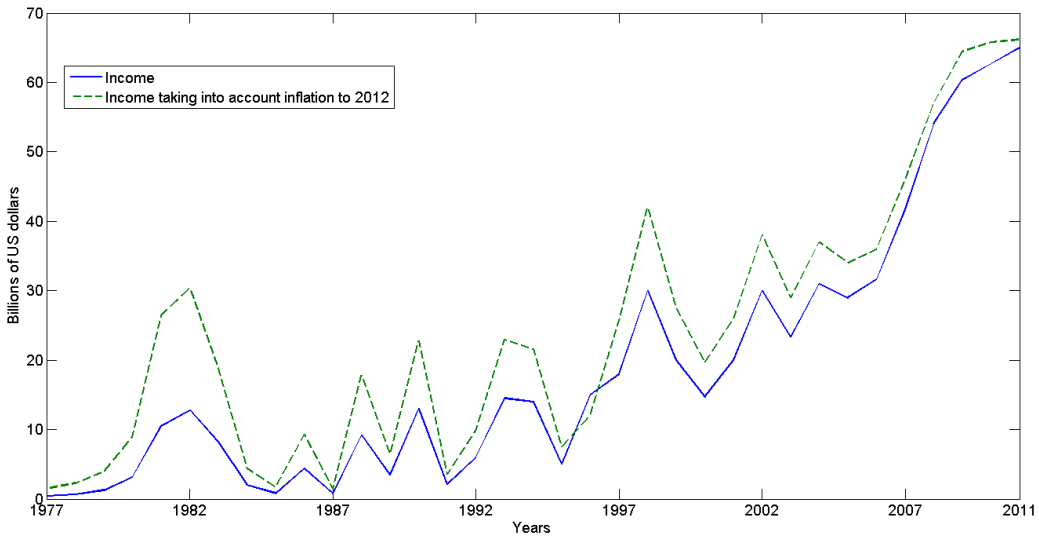

Figure 1.1: Revenue estimates of the digital game industry.

Many modern video games have non-linear narratives [2] [4] which implies that the players can perform the tasks in many different orders. Usually the game story is conveyed through cut-scenes: videos that contain dialogue between the player character and non-playable characters (NPCs). The non-linear narratives imply that the game writers cannot assume that tasks have been performed in a certain order when writing the script for a cut-scene. This results in need for the game writers to create many alternate versions of the script. This can have the cascading effect of requiring more voice acting and more video editing, as there can many different versions of the cut-scenes. This implies a higher cost for game studios and since the game industry continuously evolves it is possible that games will have increasingly non-linear narratives, making it unfeasible to use cut-scenes to convey the story. Another problem with cut-scenes is that they are not interactive, which can result in player boredom [5]. A proposed solution is replace cut-scenes with NPCs that the players can interact with using typed natural language (such as English). This is similar to existing computer programs called chatbots, which simulate conversation. The problem with chatbots is that they require a lot of work to develop as can be seen by a famous example of a chatbot called the Artificial Linguistic Internet Computer Entity (A.L.I.C.E), which has been in development for 18 years [6]. As such it would be impractical to create a chatbot per NPC, as games can have a large cast of NPCs. The proposed research is to see whether the stock responses of single chatbot can be transformed using simple rules to convey the different NPC personas. The chatbot would not have the scope of a chatbot like A.L.I.C.E, which can answer almost any question in any context, but rather focus on the context of the game. The research resulted in the development of an NPC model (which incorporates mood and dialect) to transform the stock responses and authoring tools for creating the general chatbot and NPC models. The goal is for these tools to be simple to use, so that non-technical people, such as game writers, can use them. The prototype system was created and a test game was used in conjunction with surveys to perform qualitative analysis, which indicated that it is possible to transform the stock responses of single chatbot to convey different NPC personas. Surveys were also used to test the hedonic qualities of the authoring tools. Theoretical analysis of NPC model has also been done as part of the research.

This dissertation is structured as follows:

Chapter 2: This chapter provides a general background to the research, which incorporates three fields of study, namely game development, interactive fiction (IF) and textual affect sensing. Relating to game development, the concept of a game is defined, in terms of its components (mechanics, dynamics and narrative). Interactive

fiction and textual affect sensing are briefly defined and a summary of existing methods is presented. Textual affect sensing is closely related to psycholinguistics, which is also discussed in this chapter.

Chapter 3: Existing systems in the game development, interactive fiction and textual affect sensing domains are discussed in this chapter. Two existing games, which have complex narrative structures, are discussed. Three interactive fiction systems (namely Façade, Scene-jo and Prom Week) are discussed and the complexities involved with creating these systems, as well as the perceived problems with the systems, are highlighted. The two textual affect sensing libraries used in development of the prototype, namely ConceptNet and SentiStrength, are discussed in detail.

Chapter 4: The problem of creating interactive non-playable characters is defined. The relationship between the research question and the three fields of study is also discussed in this chapter with reference to chapters 2 and 3. The exact contribution is clearly stated and the existing software frameworks used are highlighted and accredited in this section.

Chapter 5: This chapter describes the implementation details of the system, by giving a high-level overview, as well as a description of how dialect is modelled and mood acquisition. This is described using the class diagrams, flow diagrams and pseudo-code. The various iterations are the prototype system are discussed. The advantages and disadvantages of the system are also highlighted.

Chapter 6: The testing methods used and the results obtained are discussed in this chapter. The methods used to create the surveys used for the qualitative testing as well as the various metrics are described in detail. Theoretical analysis is provided to further validate the research. Interpretations of the results are also documented.

Chapter 7: The overall results of the research are summarised in this chapter. The author's contributions are re-iterated and some suggestions for further possible research are presented.

To make the dissertation easier to read, chapters start with a summary and end with a description of what is covered in the following chapter.

The next chapter gives a details description of the three fields of research being addressed: games, interactive fiction and textual affect sensing.

Chapter 2

Background of digital games, interactive fiction and textual affect sensing

The fields of research covered by this dissertation are discussed in this chapter. A brief history about game development and some taxonomy relating to digital games is presented. The field of interactive fiction is described and related to digital games. Textual affect sensing is defined and some approaches used are described.

2.1 Introduction

This research looks to improve the field of **game development**, by replacing non-interactive cutscenes with characters that players can interact with using typed natural language. This form of interactive experience falls into the field of **interactive fiction**. Another research goal is to get the characters to react to the player's mood and inferring the player's mood from typed natural language is called **textual affect sensing**. As can be seen above this and in Figure 2.1 this research incorporates three fields of research. Game development is discussed in Section 2.2, by describing some game-related terminology (Section 2.2.1) and defining the different narrative structures that currently exist in games (Section 2.2.2). Interactive fiction and agency is defined in Section 2.3. A description of textual affect sensing, as well as some techniques used, are described in Section 2.4.

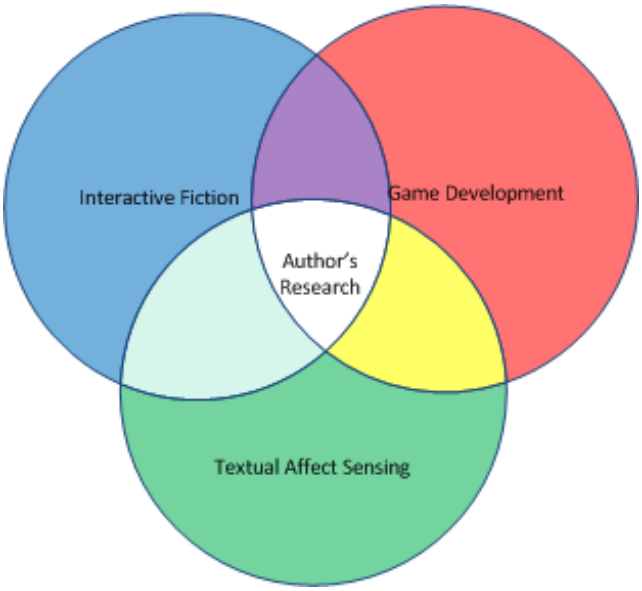

Figure 2.1: Venn diagram indicating the problem domain.

2.2 Game Development

One of the goals of the research is to create a tool that developers can use to create richer game characters, in order to immerse players and provide them with a more entertaining experience. In order to do this one needs to understand the different subsystems that make up a game, hence a taxonomy of game-related terms is given in Section 2.2.1. The main focus of the research is the storytelling aspect of digital games, which is described in Section 2.2.2.

2.2.1 Taxonomy

Game

There are many definitions of what a game is. Rogers [1] states that a game is defined as an activity that has rules, requires at least one player and there is a way to win. Another definition for a game is an activity with rules, usually containing conflict and some objectives/goals, with some form of choice [4]. As can be seen by the above definitions the exact components of a game are subject to debate but both Rogers [1] and Brathwaite et al [4] define digital games (or video games) as some form of game that makes use of a digital video screen to display the game to the player and some peripheral device to allow the player to interact with the game. This includes PC games, console games (e.g. Xbox®video game system, PlayStation®3), mobile

games (e.g. tablets, smartphones). It is in the domain of digital games that this research is being conducted.

Mechanics and Dynamics

According to Dille et al [2] the play mechanic is the core gameplay. Mechanics are defined as the rules that govern the play experience[7]. The "core" of the game is defined by Brathwaite et al [4] as the experience that the designer is trying to convey, which depends on the core mechanic (what the player can do), which in turn results in core dynamics (different play styles). From these definitions the author has come to the following definitions:

- Mechanics: the allowed actions that the player can perform.
- Dynamics: the different strategies that emerge because of the mechanics.

Player character

In non-digital games (such as board games and card games) there are game bits which allows the the players to interact with the game (interface) and some game bits may even be used to represent the player (avatar) [4, 7]. In other words the game bits are used to represent the state of the game, as it is being played. In digital games the game state can be displayed on a screen, control is performed through some peripheral device (e.g. keyboard) and the player is represented by some player character[2]. Depending on the genre and theme of the digital game, the player may control one character throughout the game, play as many different characters at different stages the game or control multiple characters simultaneously. The player character from digital games is similar to the concept of the player avatar in non-digital games.

Non-playable characters

Non-playable characters (NPCs) are characters in the game world that the player does not have direct control over and are controlled by artificial intelligence created by the programmers. NPCs include enemies, which try to hinder the player's progress, and "information-givers", with whom the player communicates. Player communication with the "information-givers" teaches the player how to play the game, informs the

player on what objectives need to be performed and/or enhance the narrative by giving the player information about the game world (such as the history of the game world). These encounters are usually non-interactive (such as cut-scenes), which creates player boredom [5], or have limited engagement in the form of conversation options (which tend to have predictable outcomes). The goal with the research is to improve these interactions and make the player feel immersed. From here on the term NPC is used to describe the "information-givers".

2.2.2 Narrative

The narrative of a digital game is the story, which includes the game world's history, NPC backstories and the progression of the plot that the player experiences. There are many ways of delivering a story in a game, both non-interactive (cut-scenes) and interactive (limited conversation options when communicating with NPCs). According to Mateas et al [8] the narrative complexity of most digital games is primitive compared to the gameplay mechanics, and this is to do with the abovementioned limited player engagement with the narrative.

In digital games there are two types of stories: writer-friendly and writer-difficult [2]. Writer-friendly narratives are linear: the player can only perform a set of tasks in one order and a such only one script has to be written, in order to convey the story. Writer-difficult narratives occur in non-linear games, where the player can perform tasks in different orders [2, 4], resulting in many alternate scripts having to be written, since the writers cannot make assumptions about which tasks have been performed. According to Dille et al [2] all non-linear games have branching narratives and there are five types of branching narratives:

- Limited branching: the player can perform tasks in different orders, but the tasks and the orders that they can be performed in are dependent on tasks that have already been done.
- Open-ended branching: the player can perform tasks in any order.
- Funneling narrative: tasks contain subtasks that can be performed in any order but the actually tasks can only be performed in one order. This allows the player freedom, while providing the writers with control of the overlying story arc and chronology.
- Critical path: the game is largely linear (player can only perform tasks in a certain order) but the player is allowed to deviate off this main task to perform tasks that do not affect the narrative.

- Nodal storytelling: these stories are dependant on what tasks have been performed or which locations have been visited, which can occur in almost any order. It is very similar to open-ended branching.

Brathwaite et al [4] define non-linear narratives slightly differently, and provide two main categories: branching and open-ended narratives. The five types of branching narratives in [2] can be grouped according to the definition in [4] as follows:

- Branching narratives: limited branching, funneling narrative and critical path.
- Open-ended narratives: open-ended branching and nodal storytelling.

This is illustrated in Figure 2.2. The abovementioned non-linear game narratives create problems for game writers since many alternate scripts have to be created to deliver a story that is dependent on the tasks performed. The goal of the research is to create a tool that allows for the relatively easy creation of interactive NPCs, independent of the narrative structure of the game and hence speed up the writing process. The goal is also to increase the narrative complexity of games by replacing non-interactive storytelling elements (such as cut-scenes), with interactive NPCs. This falls into the realm of interactive fiction, which is defined more clearly in Section 2.3.

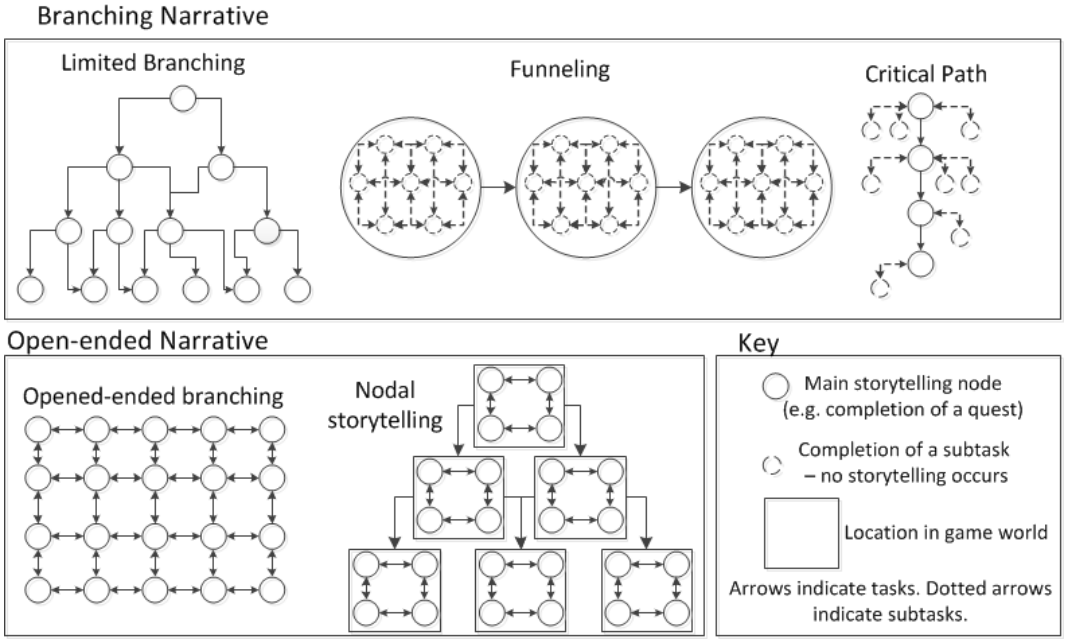

Figure 2.2: The different formats of non-linear games.

2.3 Interactive fiction

Interactive fiction has many synonyms: interactive drama [9, 10, 11], interactive storytelling [9] and interactive experiences [12], to name a few. There are also many definitions of what interactive fiction is. It is stated in [9] that interactive dramas are systems in which the audience experience the story as interactive participants. The users may be first person protagonist but this is not an explicit requirement of an interactive fiction system [9]. Some works of interactive fiction make use of artificial intelligence (AI) techniques, to manage the way the story is generated, and some do not (e.g. hypertext fiction) [9]. The focus of interactive storytelling is audience experience [9], this includes building worlds with character and story [10], where the virtual worlds are dramatically interesting and contain computer-controlled characters, with whom the user can interact [11]. From [12] the design focus when creating interactive experiences is agency, immersion (a player feels like an integral part of the story) and transformation (a player can experience different aspects of the story), where agency is the most important. Agency is defined in [13] as the satisfying ability of the player to make meaningful decisions, which have tangible effects on the play experience. It is not that the players have an unlimited number of actions at their disposal but rather there are context-based actions that can be performed, which have a very real impact on the experience [13]. The reason agency is the main focus when designing interactive fiction is because it allows the player to have the most substantial influence on the experience and requires the system to dynamically generate the story based on the player's unpredictable actions [12]. Examples of interactive fiction systems are discussed in Chapter 3.

2.4 Textual Affect Sensing

Textual affect sensing is the field of study of determining a user's mood from written text [14]. There are many other terms which are used synonymously, despite there being minor technical differences between them, such as affective computing [15], sentiment strength detection [16], affect detection [17] and mood classification [18]. Despite the psychological differences between the terms sentiment (weightings of positive, negative and neutral), emotion (specific feeling such as disappointment) and mood (experience of the emotion) [16], these terms are used synonymously in this dissertation.

The papers [15] and [19] outline the main approaches used in the field of textual

affect sensing:

Keyword spotting looks at the occurence of obvious keywords (e.g. anger). The major downfall with this approach is that it does not take into account the syntactic structure of a sentence but rather looks for the obvious keywords and in general, the recognition of negation (e.g. not) and intensity modifiers (e.g. very) is handled poorly. Keyword spotting relies on each keyword to be in a list of emotive words and generating these lists can be an ardous task. The field of psycholinguistics simplifies this task and two well-known examples of lists of emotive words are Affective Norms for English Words (ANEW) [20] and Linguistic Inquiry and Word Count(LIWC) [21]. Another problem with this approach is that, depending on the context in which a word appears, the word can have a positive or negative connotation. An approach developed to deal with this is called lexical affinity, where keywords have probabilistic values associated with them being negative, positive or neutral (e.g. accident has a 75% chance of being negative, a 5% chance of being positive and a 20% chance of being neutral). Lexical affinity is still a naive approach as it subject to poor performance under negation or intensity modification.

Statistical methods use machine learning techniques with a large set of annotated text, called a corpus, as training data. The strength of statistical methods is that the affect of different keywords, punctuation and co-occuring words can be taken into account. The major disadvantage of statistical methods is that the require large sets of input texts, and are therefore only accurate on a paragraph or page level. An example of a statistical method is Latent Semantic Analysis (LSA) [15].

Domain-specific models are proprietary and created for a specific application or a specific group of users. As such, they generally perform poorly on general text.

Knowledge-base approach.A knowledge-base is software system that consists of a nodal network of assertions made about the real world. These assertions are usually gathered through surveying a large user group (opinion mining). Some knowledge-bases have built in affect sensing algorithms and others require developers to create addition code to perform affect sensing.

The two textual affect sensing frameworks used, namely ConceptNet [22] and SentiStrength [16], are discussed in more detail in Chapter 3.

2.5 Conclusion

Some background of the three fields of study is given. Game development is described in terms of some definitions (defining a game, mechanics and dynamics, player character, NPCs). The domain of interactive fictions is introduced and the definition of agency (the main focus of interactive fiction) is given as the rewarding ability the player has to tangibly affect game play through choices. Textual affect sensing is described and four techniques are described: keyword spotting, statistical methods, domain-specific models and the knowledge-base approach.

This chapter has given a brief overview of the problem space. Examples of existing systems in the three fields of research are given in the following chapter.

Chapter 3

Review of Existing Systems

In this chapter some examples of non-linear games, interactive fiction systems and textual affect sensing frameworks are described. This information is presented to contextualise the research and aspects described below will be referred to in the following chapter in order to motivate the research.

3.1 Introduction

The research presented in this dissertation is strongly linked to three disciplines namely game development (with focus on the narrative aspects of video games), interactive fiction and textual affect sensing. While these have already been defined in the previous chapter, it is necessary to contextualise the research by presenting systems that currently exist in these fields. Two digital games, namely Rage and Mass Effect, are described in Section 3.2 and perceived strengths and weaknesses of the narratives of these games are presented. Three interactive fiction systems are described in terms of how they function in Section 3.3. Two textual affect sensing frameworks were used to create the prototype system (a result of the research) and are described in Section 3.4.

3.2 Non-linear Games

Two non-linear games are described in this section. Rage was researched through playing the Xbox®video game system version of the game for approximately twenty hours. Mass Effect was researched by watching the documentaries that are included

in the Special Edition Box Set of the game and by playing the Xbox®video game system version of the game for approximately four hours.

3.2.1 Rage

Rage is a first person shooter (FPS) game released by id Software in 2011 [23]. The game is set in a post-apocalyptic world, where a meteor strike has scattered the remaining survivors into settlements. The player takes a role of a survivor, who was one of many who were cryogenically frozen prior to the meteor's impact so that humanity could continue to survive after the impact. The player performs tasks for the town people at various settlements in exchange for various rewards, including money and survival gear. The player has to face a variety of hostiles, including bandits, mutants and "The Authority" (the power-hungry government that rules the new world). The game has a critical path narrative, as there are multiple quests which make up the main storyline. These quests are sometimes divided into open-ended subquests (that can be performed in any order) or linear subquests (which have to be performed in a certain order). There are also sidequests which reward the player with money or items but do not have a significant impact on the story of the game. The story of the game is conveyed in multiple ways:

- NPCs give the player quests and also give explanations of why the player must do the quests (hence the player is able to piece together the plot of the story by listening to the NPCs).
- NPCs randomly talk to the player directly, without giving quests, but conveying information about the game world, events or the player character.
- The player overhears conversations between NPCs about the game world, events or the player character.

The problem with these encounters is that they are done in a non-interactive way: the player simply sits passively and listens to what he/she is told, this creates player boredom and since player control is suspended agency is broken (see Chapter 2 for a definition of agency). If the player revisits an NPC, who expects the player to do something, then the NPC repeats verbatim what he/she has previously said. While it is necessary for the player to get the same information, repetition of instructions verbatim is unrealistic. This is a problem since the dialogue is set and could easily be fixed by having a selection of alternate lines (which convey the same information but are phrased differently) to randomly choose from. The problem with this solution is that it would require the game writers to do more work since they would have

to create alternate ways of saying the same thing. The fixed conversations in Rage cause repetition and lack of believability.

3.2.2 Mass Effect

Mass Effect is a role-playing game (RPG) released by Bioware in 2007 [24]. Mass Effect is set in the future, where space travel is the norm and intergalactic allegiances exists (human beings are allies with certain alien races). The game has a branching narrative structure. The player assumes the role of a captain on a spaceship and is able to specify which places the spaceship visits and which worlds his squad explores, although some are mandatory for plot progression (main quests), others act as sidequests that enhance the player experience by exposing the player to environments, enemies or rewards the player will not encounter anywhere else. This gives the player the opportunity to make meaningful decisions (in terms of risk versus reward) and this gives the player a sense of agency. Mass Effect has an interactive dialogue system that allows the player to actively participate in conversations. Instead of choosing a response and hearing the player character repeat the chosen response, the player makes emotional decision (e.g. the player may choose to "Reprimand Private" which will result in the player character saying "How dare you compromise the mission"). A screenshot is given in Figure 3.1. This interactive system allows the player to determine what information they access because the player chooses how to interact, this results in meaningful decisions and player agency. The problem with the interactive dialogue system is that the dialogue is predefined and seems to be stored in a tree-like structure and nodes can be re-visited resulting in repetition of what has already been said. This repetition breaks believability and could be solved similarly to the method mentioned in Section 3.2.1 but would have the same shortcomings.

3.3 Existing Interactive Fiction Systems

3.3.1 Façade

Façade is a one-act interactive drama, created as a research experiment in the domains of art and artificial intelligence (AI), during which the player acts as a long time friend of a married couple called Grace and Trip [25]. The player, who can choose their name from list (and implied gender), is invited to the couple's house for drinks:

Figure 3.1: A screenshot of Mass Effect's interactive story.

a situation that turns ugly [25]. The player interacts with the couple from a first person perspective using typed English and context-based actions and the interactive drama reacts to these acts appropriately [12]. Initially Façade tests the player's affinity to the other characters ("who does the player favour?") and then the player is forced to become a therapist, causing Grace and/or Trip to realise their own faults and problems [12]. Façade consists of the following subsystems [12]:

- Characters' actions are scripted using A Behaviour Language (ABL) [11].
- A drama manager that manages the dramatic beats (see below).
- A rule-based system for translating player actions and textual input into discourse acts.
- A C++ OpenGL based animation engine for rendering the drama to the screen.
- A sound engine that handles the music and the dialogue.

Dramatic beats

Façade's narrative is sequenced into dramatic beats, each of which consists of between 10 and 100 joint dialog behaviours, and only one beat is active at a time [12]. Each joint dialog behaviour is created using ABL (which defines the characters' active behaviour trees) [11] and is typically a few seconds long [12]. Beats are selected using the drama manager, based on topics of conversation and player interaction with objects in the scene [12]. Topics of conversation have three layers allowing the

player to get more detail about a topic if he/she refers to it more than once in the conversation [12]. The drama also handles topics that are not known by creating general deflections to try get the player on track [12].

Drama Manager

The drama manager (DM) or beat sequencer selects which beat to play out and, therefore, automatically makes the high level plot decisions. There are 27 beats in Façade and typically 15 are chosen during a single playthrough [25]. The beats are selected using preconditions (causal dependence), weights, priorities and the current tension levels [25]. The drama manager also provides Grace and Trip with long-term autonomous behaviours (such as pouring a drink), which are longer than joint dialog behaviours but give the illusion of intelligence and believability [12]. Beats can be interrupted and the drama manager handles this by swapping to a different beat, even if the current beat has not finished playing out [12]. Once the new beat has run to completion the previous beat is resumed.

The player can perform actions (context-based and textual inputs) that result in a certain game STATE. A complex AI entity called the drama manager manages which dramatic beat is selected based on the game STATE and the story memory (which beats have already taken place and current tension levels). The currently selected dramatic beat affects what Grace and Trip say and how they act. This results in an ineractive drama experience for the player. This is a sophisticated and powerful solution but is far too complex for a non-technical person, such as a game author, to use.

3.3.2 Prom Week

"Prom Week" is comprehensively described in [26]. A short summary of that description follows.

Prom Week is an interactive fiction game that models social interactions as gameplay. A player takes the role of a high school student during the week leading up to the prom. There is both causal gameplay elements (see what happens next) and strategic gameplay (manipulate the social environment in order to achieve specific goals for the player's character). The player is able to select one of eighteen high school students, each of which has specific social gameplay goals (e.g. transforming the nerd into the

Prom King). This is done through social exchanges: actions that the player performs on another student to change their relationship. The student has ability to accept or reject the player's intentions, based on the following characteristics:

- Basic traits: this is the core of the student's personality and does not change (e.g. the student is shy).
- Temporary statuses: these are the emotions that the student is feeling and can fluctuate (e.g. the student is happy).
- Variable relationships: these are how the student views and is viewed by the other students and can change rapidly (e.g. the student's popularity).
- Constant relationships: these social relationship change really slowly, if ever (e.g. a student's true love).

The drama manager in Prom Week is referred to as a "social physics engine", as it is analogous to the physics engines used in most digital games but instead of controlling physical mechanics it handles the social mechanics. The "social physics engine" is called Comme Il Faut and has over 5000 rules that are based on social considerations and uses the abovementioned students' characteristics to handle the social exchanges. The decisions are deterministic in nature but Comme Il Faut has some emergent rules in order to match the complexity of social interactions (e.g. friends fight but they still remain friends). There are over 20 template dialogue scenes for each social exchange and Comme Il Faut chooses the most appropriate one, which is then converted into dialogue that is specific to the students interacting (based on the abovementioned characteristics). Figure 3.2 illustrates the mechanism of a social exchange. The problem with Prom Week is that the authors had create predefined dialogue in the form of templates, resulting in a writer-difficult narrative (see Chapter 2).

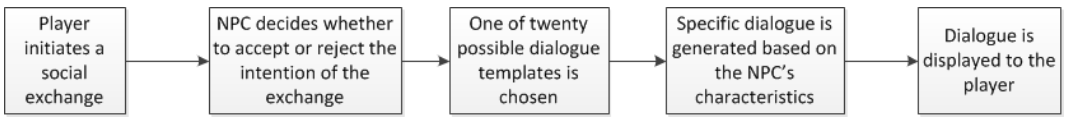

Figure 3.2: The operation of a social exchange in the interactive fiction game Prom Week.

3.3.3 Scenejo

A thorough description of "Scenejo" can be found in [27], which is summarised below.

Scenejo is an accessible authoring tool for non-programmers to create works of interactive fiction. It makes use of the open-source Artificial Linguistic Internet Computer Entity (A.L.I.C.E) chatbot[6] and Artificial Intelligence Markup Language (AIML)[28]. The difference between Scenejo and Façade is that Scenejo provides an easy-to-use tool for authoring interactive stories, while Façade provides ABL, which is extremely difficult to use. Authors are able to program Scenejo characters by writing plain AIML, using a text editor, or by using the graphical user interface (GUI) provided with Scenejo to make transition graphs. Scenejo purely supports plot generation and, as such, the emotional states of the characters must be conveyed by the author using dialogue. In Scenejo the plot of the interactive fiction is split into acts, which is made of scenes. A scene allows for different reactions and dialogues to occur over time and contains a subset of the overall AIML. The author can either have full control over the plot or can use conditions to determine scene changes and create an emergent plot. Scenejo allows for the follow conditions to change the scene:

- Elapsed time.
- User utterances.
- Miscellaneous state changes.

All scenes are made of actors and dialogue graphs. Actors can have knowledge which is dependent on the scene and permanent knowledge, which is scene independent (see Figure 3.3 for a representation of this). Dialogue graphs are made of stimulus-response elements (SREs), represented by AIML patterns (which map an input to an output). A GUI is used to create the dialogue graphs and the scenes out of SREs. The structure of Scenejo-based interactive fiction can be seen in Figure 3.4. Scenejo requires that the authors take the NPCs' moods into account and as such cannot dynamically assign mood. Scenejo uses AIML extremely verbosely and requires a knowledge-base per character per seen, resulting in an inefficient writing process.

3.4 Existing Affect Sensing Frameworks

3.4.1 ConceptNet

"ConceptNet" is described, in detail, in [22]. A summary of this description follows.

ConceptNet is a natural language understanding framework, which can analyse text in terms of context, concept and affect; and give analogies and summaries. It is based

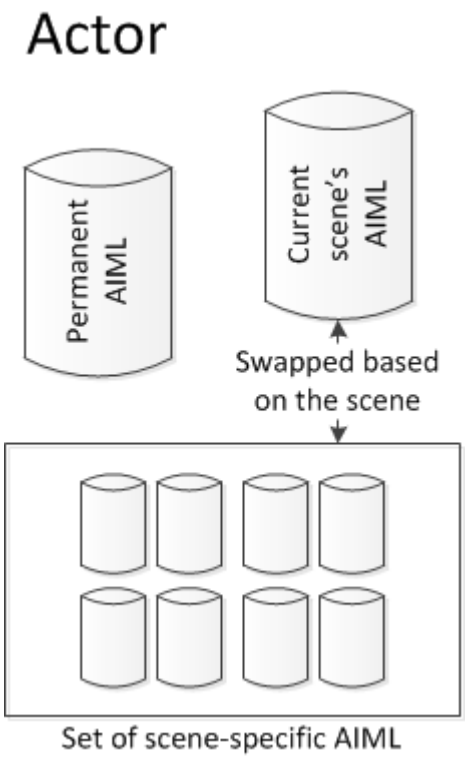

Figure 3.3: Representation of an actor's knowledge in Scenejo.

on the notion that to fully understand text common sense is required. Common sense is the millions of basic facts that most people understand because of their experiences of the world. Common sense is needed for the following reasons:

- Ambiguities in text: e.g. the word "cookies" has drastically different meanings in "My computer has many corrupt cookies" and "Chocolate-chip cookies are my favourite".
- Affect sensing: e.g. an affect sensing algorithm that does not make use of common sense can easily get the mood from "I am sad" because of the presence of the affective word "sad" but would struggle with "My dog just kicked the bucket" because of the lack of obvious emotive keywords.

ConceptNet is similar to the common sense reasoning programs, Cyc and WordNet, but it is readily available to the public in the form of a free-to-use Python library. ConceptNet was created from the Open Mind Common Sense (OMCS) knowledge-base using the automatic process shown in Figure 3.5. Assertions in ConceptNet are structured into the following groups:

1. K-Lines (1.25 million assertions): ConceptuallyRelatedTo, ThematicKLine, SuperThematicKLine.

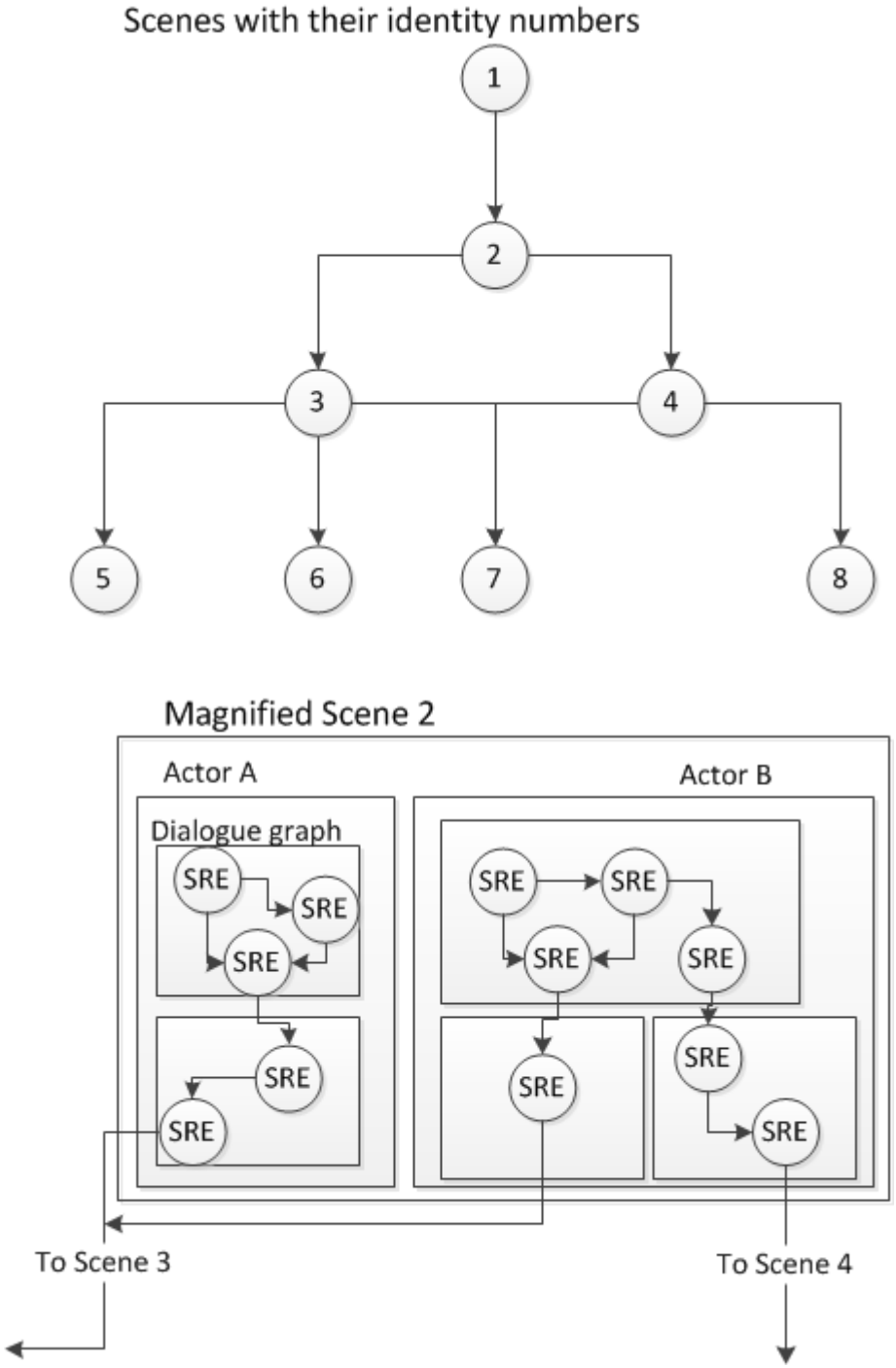

Figure 3.4: Structure of an interactive fiction plot in Scenejo.

2. Things (52000 assertions): IsA, PropertyOf, PartOf, MadeOf, DefinedAs.
3. Agents (104000 assertions): CapableOf.
4. Events (38000 assertions): PrerequisiteEventOf, FirstSubEventOf, SubEventOf, LastSubEventOf.
5. Spatial (36000 assertions): LocationOf.

6. Causal (17000 assertions): EffectOf, DesiredEffectOf.
7. Functional (115000 assertions): UsedFor, CapableOfReceivingAction.
8. Affective (34000 assertions): MotivationOf, DesireOf.

ConceptNet makes use of this assertion structure to store its common sense. Textual affect sensing is performed by rating text according to the Ekman model of emotion[29], which implies that every possible emotion can be represented by a weighted sum of the six core emotions, namely happiness, sadness, anger, fear, disgust and surprise. This is calculated by analysing the paths traversed in the assertion network in order to understand the text and makes use of the associate emotive qualities and relative frequency of the paths.

3.4.2 SentiStrength

"SentiStrength" is comprehensively described in [16]. The following is a summary.

SentiStrength is a textual affect sensing framework, which makes us of keyword spotting and statistical methods (see Chapter 2 for a descriptions of textual affect sensing, keyword spotting and statistical methods) to assess the sentiment of informal text. SentiStrength gives text a rating of positive (1 to 5) and negative (1 to 5) simultaneously, which differentiates it from other opinion mining systems which do not give both positive and negative ratings simultaneously and do not give strengths of the sentiment but rather an indication of whether it is positive or negative. SentiStength was created using MySpace user comments (which provided a large quantity of informal text) as training data sets and evaluation data set. A large list of users was selected from MySpace. This list was pruned by removing the following profiles:

- Users that are not US nationals.
- Celebrities.
- Inactive users: two or less friends or no comments.
- Abnormal users: more than 1000 friends or 4000 comments.

2600 comments were given both positive and negative ratings (on the scale mentioned above) and were used to generate a sentiment word list of 298 positive words and 465 negative words. Some words are considered both positive and negative (e.g. "miss" has a rating of +-2 since the sentence "I miss you" has associated emotions of sadness and love, which are negative and positive respectively).

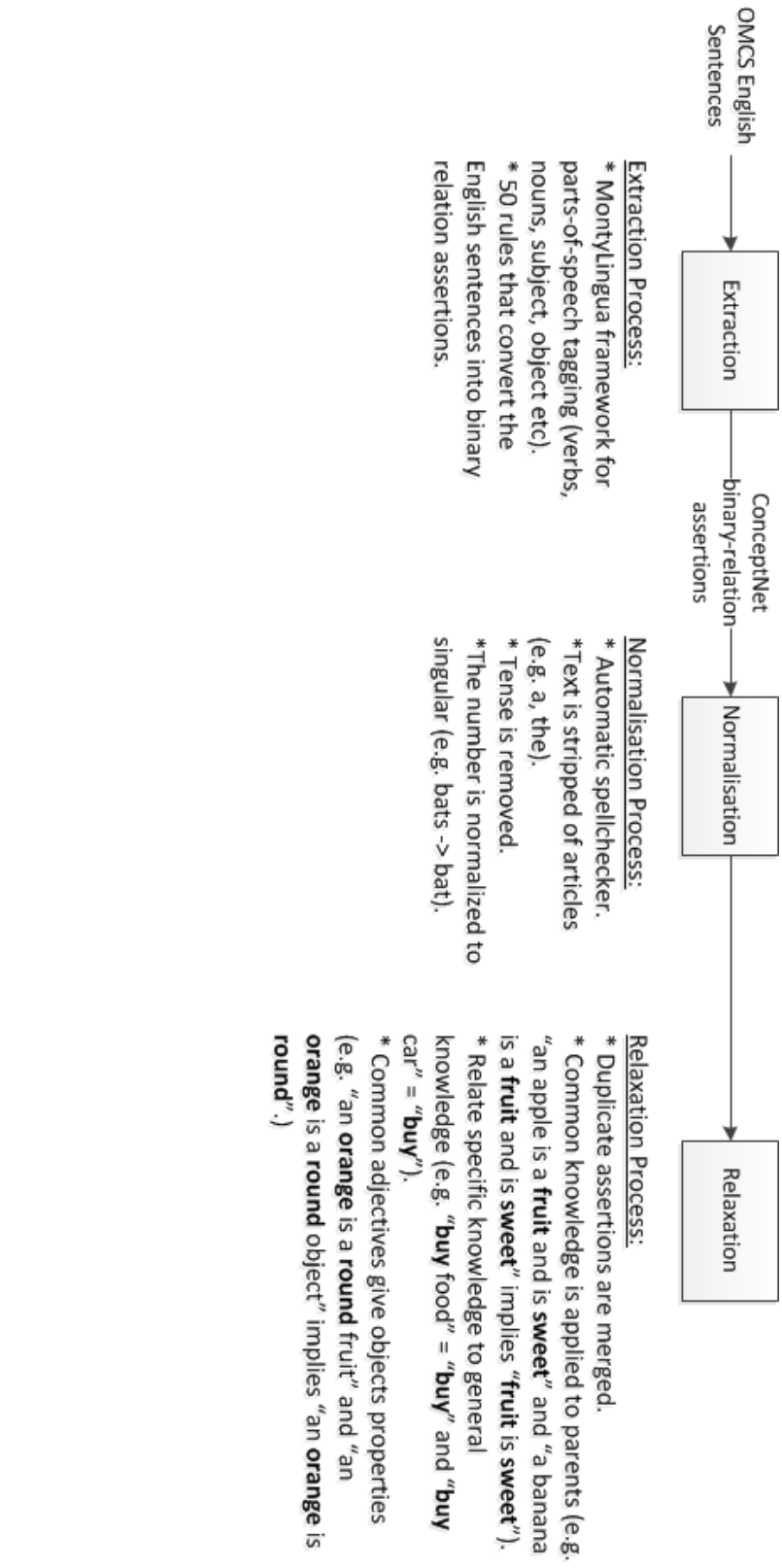

Figure 3.5: Automatic creation process of ConceptNet from OMCS.

SentiStrength consists of the following which make up its corpus:

- An optimised sentiment word list: an algorithm was applied to the abovementioned sentiment word list in order to speed up the ranking process.
- A booster word list: words such as "very" increase the strength of the sentiment.
- A negating word lists: word such as "not" invert the value of the sentiment.
- An emoticon word lists: emoticons are typed symbols, such as ":)", which are used often in informal text to convey emotion.

SentiStrength first performs spelling correction as follows:

1. Deletes repeated letters that occur more than three in a row ("helllo" becomes "hello").
2. Deletes repeated letters that rarely occur twice in a row ("niice" becomes "nice").
3. Deletes letters occuring twice in a row if the word is not an English word but deleting will result in an English word.

SentiStrength then applies keyword spotting using the abovementioned lists. Repeated letters are sometimes used in comments to show emotion and SentiStrength takes this into account by boosting the strength if there are more than one repeated letters (e.g. "helloo" would be considered incorrect spelling by SentiStrength and "hellooo" would boost the emotion). Negative emotion is ignored in questions (e.g. "Are you angry?" is seen as a neutral sentence despite the presence of the negative keyword "angry"). SentiStrength has the weakness of succumbing to ambiguities in the text, this is because keyword spotting is semantically weak (as mentioned in Chapter 2) and does not take into account the overall affect of certain phrases such as idioms, which could effect the accuracy of the ranking. The test data was created using five human (female) judges, who gave 1041 comments positive and negative ratings. Two of the judges' rating were rejected, because one gave inconsistent results and the other gave much higher positive ratings than the others. The remaining three judges ratings were averaged and used as a "gold standard". SentiStrength's ratings were then compared to this "gold standard" and other affect sensing techniques and it was calculated that SentiStrength has an accuracy of 60.6% for positive ratings and 72.8% for negative ratings.

3.5 Conclusion

Examples of existing systems in the fields of digital games, interactive fiction and textual affect sensing are described above. The two non-linear games Rage and Mass Effect suffer from repetition since the dialogue is predefined, but Mass Effect's narrative is better since it is interactive (providing the player with a sense of agency). The interactive drama, Façade, allows the player to interact using natural language but has the drawback of an extremely complex AI system called the drama manager, which manages the story. Another problem with Façade is that it was extremely complex to author and non-technical people would struggle to create their own stories using the Façade approach (using the ABL programming language). Prom Week, an interactive fiction game, has a "social physics engine" - a drama manager that handles all the player's social actions. The problem with Prom Week is that the dialogue is all predefined in dialogue templates, which made authoring a labourious task. The interactive fiction authoring tool, Scenejo, allows non-technical people to create interactive fiction creating a nodal network of scenes that result in the overall plot. While it does allow for some variable narrative (since it makes use of chatbots to model dialogue) Scenejo requires the moods of the NPCs to be handle directly by the authors and is fairly work intensive to use. ConceptNet is a natural language understanding framework that uses assertions about the real world to "understand" text and provides textual affect sensing capabilities. SentiStrength judges the sentiment of formal text using a corpus created from MySpace comments using machine learning techniques.

Existing non-linear games, interactive fiction systems and textual affect sensing frameworks are described in this chapter. The following chapter contains a detailed description of the problem, with reference to the existing systems presented in this chapter, and descibes the research question.

Chapter 4

The Problem of creating Interactive Non-Playable Characters

The problem of creating interactive NPCs is specified in detail, by analysing the weakness of existing systems through research. This research informed a research question, which in turn resulted in the requirements of the prototype software system. The main focus on the research is creating an NPC model of dialect and mood that can be used to translate the stock responses of a single chatbot to convey different NPCs' personas. The single chatbot acts as a central knowledge-base, allowing for the code re-use paradigm of software development to be applied to game writing. The focus of the research is not creating a new chatbot architecture or affect sensing algorithms and as such the prototype makes using of existing frameworks to perform these functions.

4.1 Introduction

The motivation of the research is given in Section 4.2, by referring back to Chapters 2 and 3. The focus of the research, including the research question, is clearly stated in Section 4.3. Section 4.4 contains the definition of the scope of the research. The methodology and conclusion are given in Sections 4.5 and 4.6 respectively.

4.2 Motivation

As mentioned in Chapter 2, there are writer-friendly and writer-difficult narrative structures in games, namely linear and non-linear narratives respectively. Non-linear narratives are writer-difficult since the player can perform tasks in different orders and as such the writers cannot assume that tasks have been performed in a certain order and end up having to write multiple scripts to convey the story differently depending on which tasks have been done. As discussed in Chapter 3, the game Rage has a non-linear narrative but the dialogue with NPCs is set (which results in repetition which decreases the believability of conversations) and non-interactive (which creates player boredom). The game Mass Effect has an interactive dialogue system but the dialogue is set and can be repetitive, which is unrealistic. Both these games required a large amount of work from the writers. The goal of the research is create a tool that allows for interactive NPCs to be created as easily and quickly as possible.

Interactive NPCs fall into the realm of interactive fiction. Three IF systems are discussed in Chapter 3 and have the following problems:

- Façade: is managed by an extremely complex AI system called the drama manager. The behaviour of NPCs is controlled using active behaviour trees using a language called ABL, making it difficult for non-technical people to create their own stories. The dialogue is predefined, which can result in predictability and unrealistic results.
- Prom Week: social interactions are managed by the complex "Social physics engine" and has set dialogue, based on dialogue templates. This made authoring the conversations a complex tasks and since the dialogue is predefined, it can be predictable and repetitive.
- Scenejo: relatively simple to use, but uses AIML extremely verbosely (knowledge-bases per character per scene) and does not automatically handle emotions (the author has to make the dialogue reflect the NPCs' moods).

The goal of the research (presented in more detail in Section 4.3) is to make it easy and quick to create interactive NPCs, which communicate with player using procedurally-generated dialogue (since the variability in dialogue should give an illusion of intelligence and realism) [30]. The NPCs should automatically perceive the player's mood and respond accordingly. The research question and the exact contribution of the author are also outlined in Section 4.3. The research results in

the creation of a prototype system, which makes use of multiple existing software frameworks, this is discussed in Section 4.4. An overview of the research methodology and the software development methodology is given in Section 4.5.

4.3 Focus of the research

The aim of the research is to create a tool that non-technical people, such as story writers for games, can use to easily create interactive NPCs. Of the existing interactive systems discussed Chapter 3, Scenejo, is the most similar to the system created as part of this research but it makes use of multiple chatbot knowledge-bases making authoring quite repetitive and laborious. To improve this, the code re-use paradigm of software development can be employed in the field of interactive fiction and hence the proposed system requires a single chatbot to be trained, regardless of how many NPCs are in the game. The generic responses of the single chatbot are transformed so that they reflect the persona of the NPC that the player is talking to. Formally stated, the research question is:

Can the stock responses of a single chatbot be transformed to convey the personas of different NPCs and thereby reduce the amount of effort required to simulate conversation with different NPCs?

4.4 Scope

As can be seen by the research question, the research focus is on creating a translation layer to convert generic responses into NPC-specific responses, and not creating the general chatbot itself. As such the prototype system makes use of an existing chatbot framework and affect sensing frameworks. Another important aspect to mention is that the focus of the research is on emergent dialogue and not necessarily on conveying a story, hence no drama manager was created. Emergence means simple rules that result in complex, unpredictable behaviour [31]. An example of emergence is Chess: the rules are extremely simple but the game can be played in so many different ways (different strategies). Another reason for the lack of drama manager in the proposed system is that the it should work regardless of the narrative structure and as such provides a basic conversation framework that developers could combine with their own specific drama managers. For example, the proposed system could be used to generate the dialogue for a system like Prom Week so that dialogue would

not have to be pre-scripted, but the authors would still have to combine the proposed system with the "Social physics engine". The process followed in answering the research question is described in Section 4.5.

The scope of the research is modelling NPCs' dialects and moods, in order to create a translation layer which converts the responses of a general chatbot into NPC-specific responses. The scope does not include creating a new chatbot architecture or affect sensing algorithm and as such these tasks are performed by existing frameworks as can be seen in Figure 4.1. Affect sensing is done using SentiStrength and ConceptNet (which are described in detail in Chapter 3). The general chatbot uses PyAIML [32], a python-based AIML interpreter. AIML is used in the chatbot A.L.I.C.E and provides a method of mapping user inputs to response and provides functionality for synonyms and multiple responses.

As mentioned in Section 4.2 and Chapter 3 Scenejo, Prom Week and Façade suffer from authoring complexity and predefined, predictable dialogue. The strengths of these systems over the proposed system is that they offer the author superior control over the story. In contrast, the proposed system simply procedurally generates dialogue and developers using the system would have to create the software to manage the story. The envisioned strength of the proposed system is the emergent dialogue, ease-of-use and efficiency of writing (because of the code re-use paradigm). The scope is summarised in Figure 4.2.

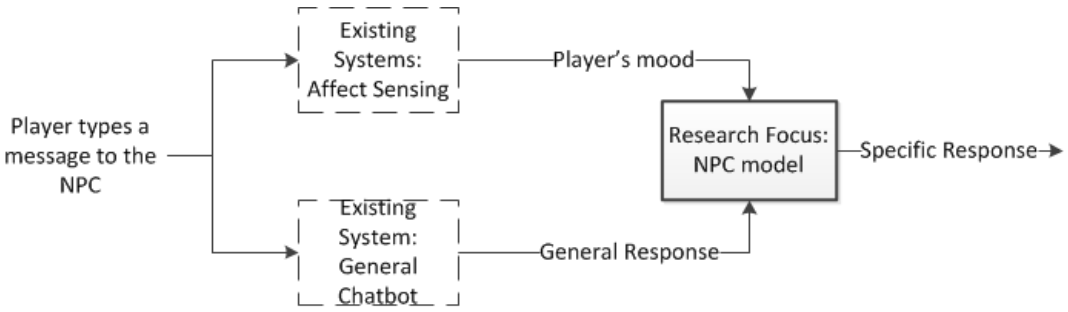

Figure 4.1: The focus of the research and the use of existing systems.

4.5 Methodology

The research process went follows:

1. A prototype software system was designed and created.

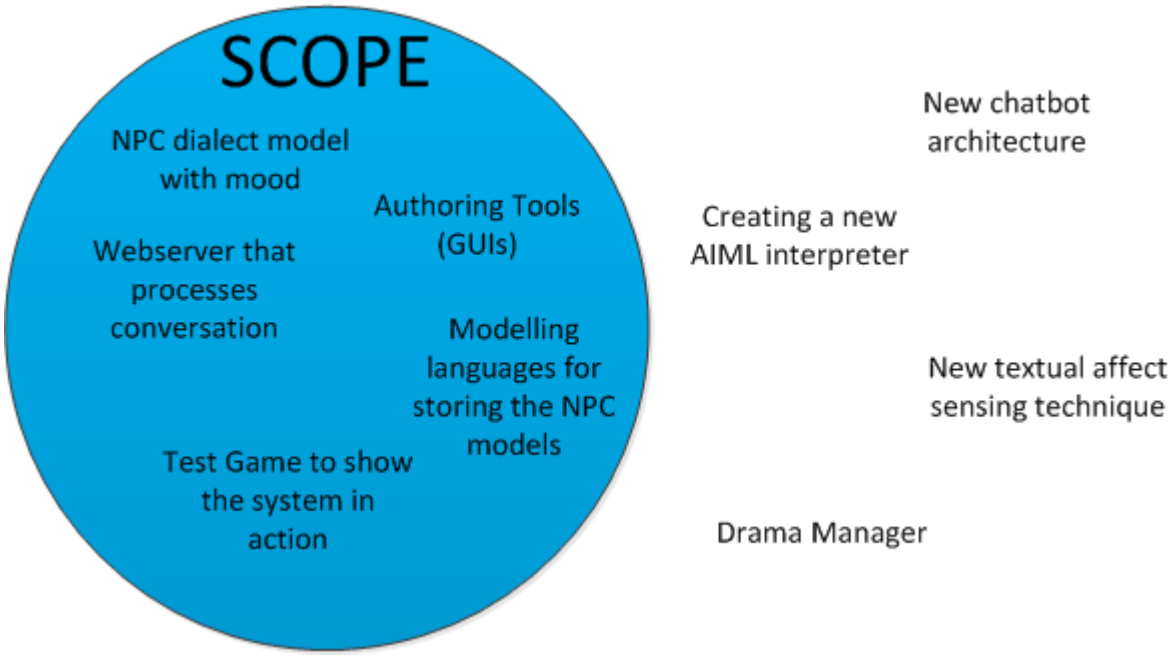

Figure 4.2: Summary of the problem scope.

2. Qualitative testing was peformed via user group surveys, to gauge user experience (from interacting with a test game) and to assess the hedonic quality of the authoring tool.
3. Quantitative testing was performed using automated metrics, gathered from user interaction with a test environment.
4. Theoretical analysis of the emergent rules was done by deriving equations to describe the variability of the rules.

The fact that user-based testing is done meant that ethics approval is required. The problem with using a test game to assess the proposed system is that it does not test how well the proposed system works but rather how well the authoring tools were used. This is offset by using automated metrics and theoretical analysis. For more details on the testing process see Chapter 6. The requirements engineering process allowed for the weaknesses of existing systems (mentioned in Section 4.1) to be identified and converted into system requirements. These requirements are shown in Table 4.1. The software prototype was created using the iterative development paradigm [33].

Table 4.1: The requirements and the corresponding solution for the prototype software tool.

Requirement	Solution
NPCs should respond to player's input of typed natural language such as English.	The tool uses of an existing chatbot framework, since the chatbot is not the focus of the research.
Game writers should be able to create NPCs quickly.	The tool makes use of one chatbot, forming a single knowledge-base for all the NPCs. This is to increase the speed of the writing process by ensuring that common-knowledge is not rewritten. This is analogous to the code re-use paradigm of software development.
The software should be easy-to-use.	A graphical user interface is provided to game writers for authoring.
The dialogue should be emergent, instead of predefined.	The generic responses of the single chatbot are translated to match the NPCs' personas using simple probabilistic rules.
NPCs should detect the player's mood and respond in a fashion determined by the game writers.	The system makes use of textual affect sensing frameworks.

4.6 Conclusion

Games usually have non-interactive narratives and even when they have interactive narratives the dialogue used is predefined, resulting in repetition and thererfore lack of believability. In the field of IF it is generally difficult for non-technical people to create interactive stories. Even systems, such a Scenejo, that aim to make the authoring process easier have drawbacks such as requiring authors to create knowledge-bases for each NPC, resulting in long training process for each NPC and making it impractical to make lots of different NPCs. This inspired the concept of having one chatbot knowledge-base which would take a fair amount of time to train, and having its responses translated into NPC-specific responses using a translation layer that the authors can define. The training of each NPC's translation layer should take a lot less time then training a knowledge-base for each NPC. These problems

and requirements, resulted in a research question, that was answered through the development and testing of a prototype system.

The focus of the research, the research question and methodology are discussed in this chapter. The following chapter contains a description of the implementation of the prototype software system.

Chapter 5

Prototype System Implementation

The prototype system consists of a Python-based webserver that parses markup files which are created by game writers using the C#-based GUIs. The game acts as a client and communicates with the webserver using URL-encoded requests. The system makes use of three frameworks: PyAIML (general chatbot), ConceptNet and SentiStrength (textual affect sensing).

5.1 Introduction

The implementation of the prototype system consists of many components. An overview is given in Section 5.2, which describes the client-server architecture of the system. The communication between the client and the server is described in Section 5.3. GUIs have been created to make it easier to create NPCs: the GUI for creating the general chatbot and the GUI for creating the NPC models are described in Sections 5.4 and 5.7 respectively. The models of dialect and mood are described in Sections 5.5 and 5.6 respectively.

5.2 High Level Overview

The prototype system has a Python-based web server which hosts the general chatbot and performs the NPC transforms. The reason for hosting on a webserver is so that the models for the communication do not take up local storage space and processing power. Whether the storage space or processing is significant is not the purpose of the research and as such has not been investigated further - this is just a proposed

methodology. The general chatbot was created using PyAIML, a Python-based AIML interpreter [32] and the chatbot's knowledge-base can be trained using the GUI to quickly create AIML files which can be uploaded to the server (see Section 5.4). A client (e.g. a game) can communicate with the webserver using Hypertext Transfer Protocol (HTTP) and it responds with a Hypertext Markup Language (HTML) document (see Section 5.3 for more detail on the communication protocol). The client has the responsibility of storing the game and NPCs' states (e.g. NPCs' moods) as games can have many players and it would result in less strain on the server to distribute state logic to the client instead of storing it in one central location. The system has been designed such that developers can integrate the NPC translation layer functionality with their own chatbot systems (see Section 5.3 for more detail). The NPC translation layer is created using a GUI (see Section 5.7) which creates Character Modelling Language (CML) files (see Section 5.5) which are uploaded onto the server. CML and AIML files are loaded into the server's Random Access Memory (RAM) via XML manifest files. The training GUIs were created using Visual C#. Figure 5.1 shows the system components.

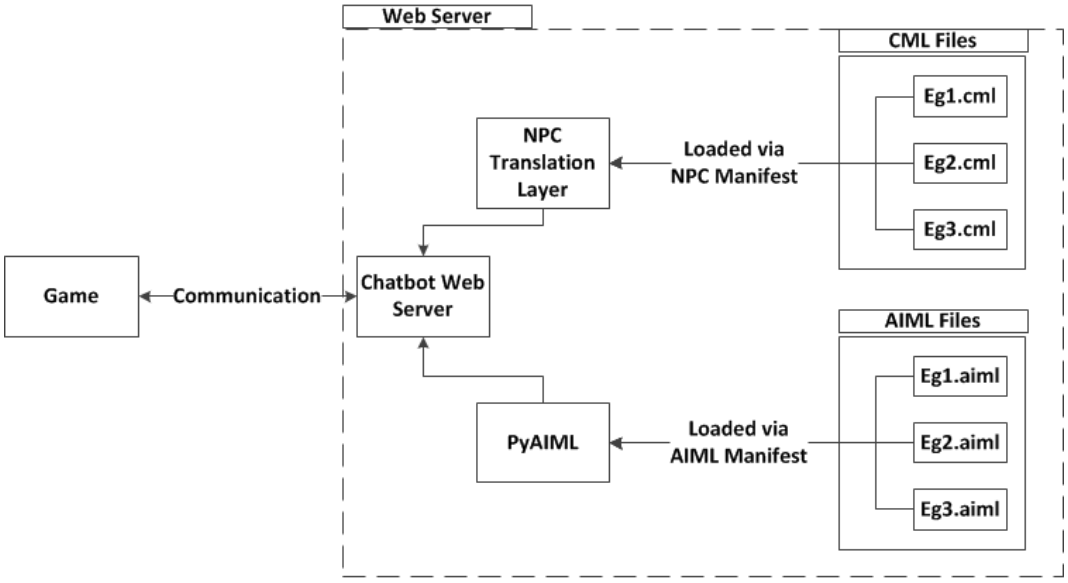

Figure 5.1: The prototype system's architecture.

5.3 Communication protocol

The prototype uses a client-server architecture, where the game runs as a client and communicates with the server. The server does not provide functionality to store NPC or game state as many instances of the game could be running simultaneously.

As such the client (the game) should store the game state such as NPCs' moods and player progress and communicates with the webserver to request certain tasks to be done such as calculating an NPC's mood or response. HTTP requests are structure such that paramaters form a URL structure (e.g. /A/B/C gives the parameters "A", "B" and "C"), the parameter order is critical and the requests are URL encoded (e.g. a single whitespace becomes "%20"). The valid requests are given in Table 5.1.

Table 5.1: HTTP request structure for communication between the game (client) and the webserver.

Request	Function	Example
/NPC_ID/conversation /MoodValue/message	Get a stock response to "message" from the general chatbot and convert it to a specific response for the NPC with "NPC_ID" and a mood of "MoodValue".	/Bob/conversation /Anger/Why%20are %20you%20cross?
/NPC_ID/translate /MoodValue/response	Translate "response" into a NPC-specific response using "NPC_ID" and "MoodValue". This functionality is provided so that NPC translation layer can be integrated with a different chatbot system. For example, a developer may feel that their own proprietary chatbot system is better at providing generic responses, but still wants to use the NPC translation layer to generate NPC-specific responses. This can be done using translate.	/Bob/translate/Sadness /Proprietary%20 General%20 Chatbot%20Response

Table 5.1: HTTP request structure for communication between the game (client) and the webserver.

Request	Function	Example
/NPC_ID/debug /MoodValue/message	Get the general and specific responses and output both of them. This also outputs all debug information.	/Bob/debug/Happiness /Why%20are%20you %20cross?
/NPC_ID/changemood /string	Run textual affect sensing on "string" and calculate the new mood of the NPC with "NPC_ID". Use this function to update a NPC's mood in client.	/Bob/changemood/You %20are%20a%20 moron
/message	Get the general chatbot's response to "message".	/Why%20are%20you %20cross?

5.4 General Chatbot GUI

The prototype system makes use of PyAIML, a AIML interpreter based on the A.L.I.C.E chatbot. The general chatbot uses AIML to store its knowledge-base. AIML is fairly difficult-to-learn XML based language and as such it was decided to create a GUI that would allow non-technical people to easily create AIML files without understanding the language. AIML is a fairly complex language and has quite a lot of features but since the general chatbot is not the main focus of the research only a subset of the AIML functionality is visualised in the GUI. In AIML there are patterns: phrases (which can include wildcard symbols that indicate the position of the phrase in the sentence) that trigger certain events when detected in the input string. Two of these events are:

- Respond: a random response is chosen from a set of possible responses.
- Symbolic reduction: the pattern is linked to another pattern, allowing synonym lookups.

For the sake of the GUI a slightly different nomenclature to AIML has been chosen so that it is easier to understand and also the representation is quite different: the AIML is parsed and converted into a C# Tree GUI element. The first level node of the tree is the keyword node (a pattern that does not result in the symbolic reduction but rather in a response). The keyword node has two children, which are the headings "Synonyms" and "Responses". The "Synonyms" node has all the synonyms (all the patterns that make use of symbolic reduction) as children and the "Responses" node has all the possible responses to the keyword. This tree structure is illustrated in Figure 5.2. The user is able to select a node on the tree by double left clicking and this determine which mode of operation the GUI is in, which can be one of the following:

- Keyword Mode: a keyword node has been selected. The user is able to create a new keyword or edit, delete or visualise an existing keyword. Visualizing a keyword generates an image which represents its relationship with its synonyms and responses (see Figure 5.3 for an example).
- Synonym Mode - Add New: a "Synonyms" heading node has been selected. The user is able to add a new synonym (indicating where the synonym shall appear in the sentence or if it must be detected simply as is). The system can also suggest synonyms of the keyword, which the user can select to add. The synonyms are generate automatically using the NHunspell .NET library [34]. Figure 5.4 is a screenshot of this mode of operation, showing the system generated synonym list.
- Synonym Mode - Add/Edit/Delete: a synonym node has been selected. This provides the same functionality as "Synonym Mode - Add New" but also allows the user to edit or delete the selected synonym and the system generate synonym list is based on the selected synonym and not the keyword.
- Response Mode - Add New: a "Responses" heading node has been selected. The user is able to add a new response.
- Response Mode - Edit/Delete: a response node has been selected. The user is able to edit or delete the existing response.

AIML files are then uploaded onto the server. On startup of the Python-based webserver the AIML files are loaded into memory, forming the general chatbot's knowledge-base. An AIML manifest file specifies which AIML files PyAIML parses for use in the general chatbot. This allows similar AIML patterns to be grouped into smaller files, instead of having one monolithic AIML file, which defines all of the patterns. This manifest file can be created via a GUI, as shown in Figure 5.5.

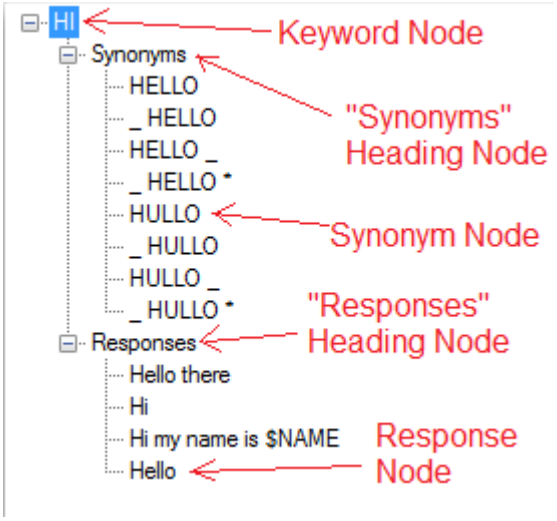

Figure 5.2: The tree structure in the general chatbot GUI.

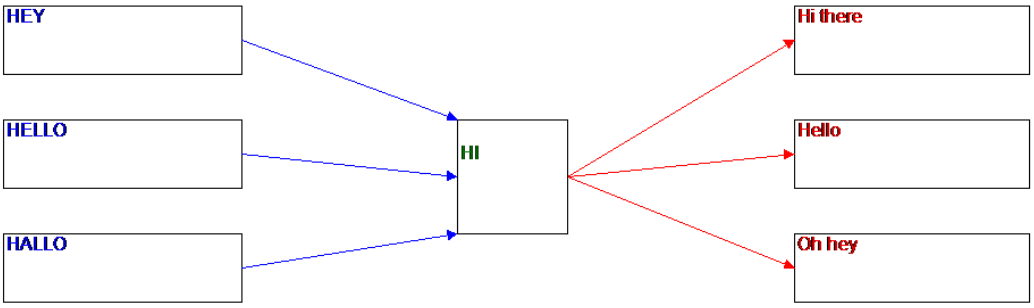

Figure 5.3: Visualization of a keyword: the synonyms (leftmost boxes) result in the keyword (centre box) which gives the responses (rightmost boxes).

5.5 Model of Dialect

The term "dialect" is used to describe how an NPC phrases a sentence (e.g.what words are used). In this dissertation "dialect" is NOT used to describe subtle differences in language due to the area the speaker comes from. The model for dialect is based on the concept of emergence - simple rules resulting in complex, unpredictable behaviours [31]. According to Hendrikx et al [30] procedurally generated behaviour can result in the illusion of intelligence and as such the procedural conversion of a generic chatbot's responses to NPC-specific responses is intended to create the illusion of intelligence. Three simple transform rules make up the dialect model, namely:

- Substitution: transforming one phrase into another. For example: cannot becomes can't.
- Prefix: a phrase which is appended to the beginning of a general response.

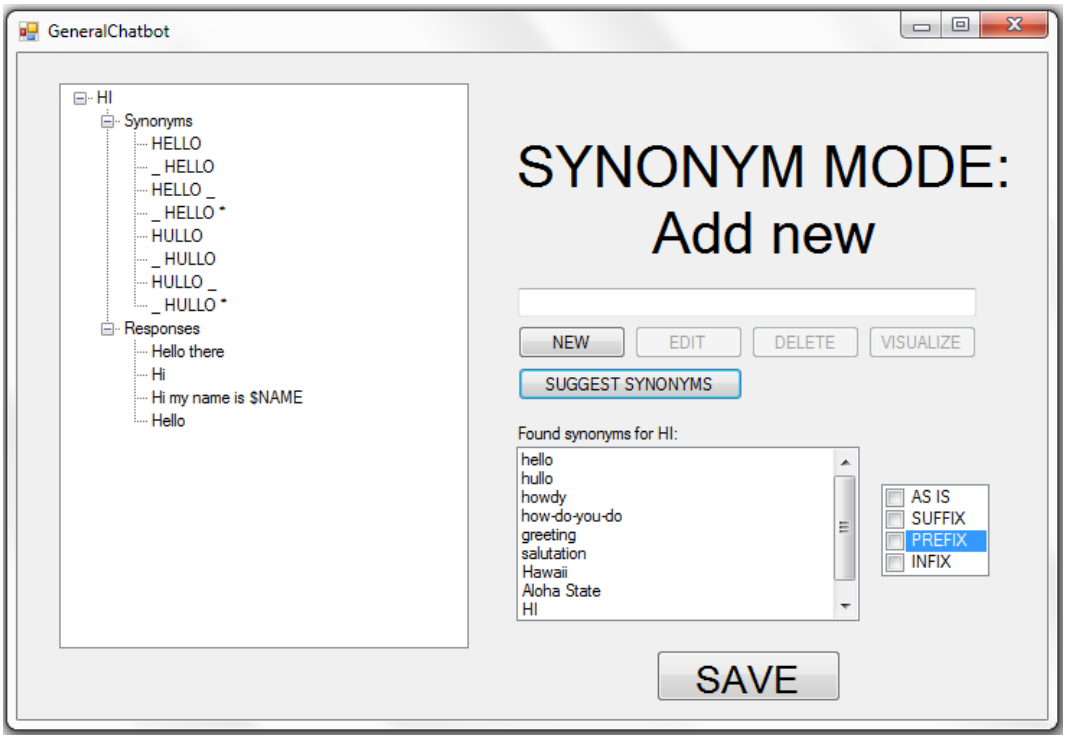

Figure 5.4: The GUI in "Synonym Mode - Add New" mode, with the system generated synonym list.

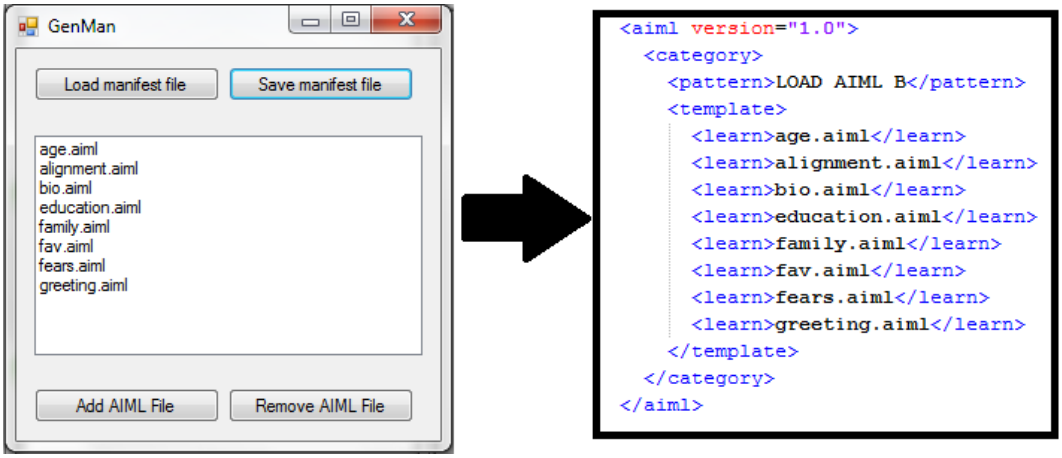

Figure 5.5: The GUI for creating the manifest file used by PyAIML to load the chatbot knowledge-base and the resulting manifest file.

- Suffix: a phrase which is appened to the end of a general response.

These rules each have a probability of occuring, which is dependent on the current mood of the NPC. Only one prefix and one suffix can be applied per response, in order to prevent stacking of prefixes and suffixes which may result in a response that does not make sense and/or is extremely verbose. Multiple substitutions can be applied per response. A basic class diagram of the dialect transform rules is given in

Figure 5.6. The prefix and suffix rule lists are randomly shuffled at the beginning of the translation process. This is to ensure that first rule in the lists is not always the same and therefore there is not a bias towards the first rule. For example if shuffling did not take place and the first prefix rule had 50% probability of occuring, then 50% of the time this rule would be applied and 50% of the time it would not. This means the rest of the prefix rules would only have 50% chance of occuring, since only one prefix rule can be applied at a time. This example also applies for suffix rules. As such, shuffling is applied so that the lists of rules are always in different orders so that one rule is not favoured over the others.

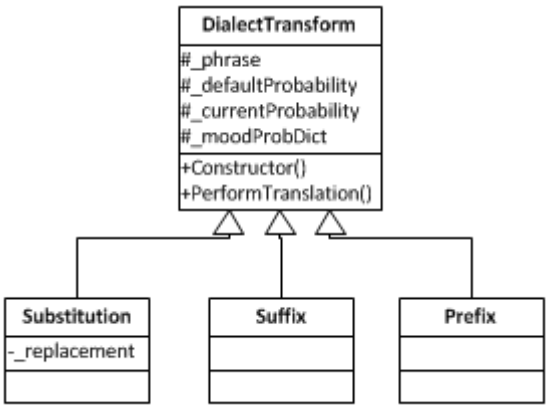

Figure 5.6: A simplified UML class diagram of the dialect transform rules.

Character modelling language (CML) was created so that authors can easily create NPCs. Included in CML is Dialect Transforms Modelling Language (DTML), which defines the above-mentioned dialect transforms and their mood-dependent probabilities. CML files can contain DTML tags for performing the dialect transforms or can reference external DTML files, allowing a single set of dialect rules to be used by different NPCs. For example, a game writer may wish to create a list of contractions and save it in “contractions.dtml”. The game writer could then make several characters that use the dialect transforms defined in “contractions.dtml”. This is provided to speed up the writing process, similar to the code re-use paradigm of software development. Game writers may wish to provide NPCs with specific knowledge such as biographical information (e.g. the NPC’s name, family and alignment). For this reason lookup transforms are provided. For example: a player may type “What is your name?” and the general chatbot may respond “$NAME”. The “$” indicates that “NAME” is a lookup transform. The NPC’s translation will then convert “$NAME” to “Bobby” if the game writer has defined this as the lookup. In traditional storytelling there are different type of characters (e.g. round, flat and developing characters) [35]. In game writing the lead game writer may wish to

define different NPCs by the set of lookup transforms that they have available to them (e.g. a round character may have forty different lookup transforms while a flat character may have only ten available lookup transforms). This functionality is provided via Lookup Template Markup Language (LTML) files. LTML files cannot be externally referenced like DTML files (mentioned above) but are simply loaded into the CML files and overloaded. This is because lookup transforms are specific to an NPC and cannot be shared (e.g. two NPCs should not have exactly the same conversion of "$NAME"). CML also models mood but this is discussed in Section 5.6. CML also provides the game writers with the ability to define "Response Not Found" (RNF) responses for the NPCs. An RNF is chosen at random either when the general chatbot cannot provide a response to the player's input or when the general chatbot response contains a lookup which is undefined for the NPC that the player is interacting with. Figure 5.7 gives the flow of the translation process.

5.6 Mood detection

As mentioned in Section 5.5 the dialect model of the NPC is mood dependent. It is, therefore, necessary to be able to rank the emotive qualities of the player's inputted text. This falls into the field of textual affect sensing (described in Chapter 2). Two existing textual affect sensing frameworks are used, namely ConceptNet [22] and SentiStrength [16] (described in detail in Chapter 3). ConceptNet ranks the text according to Paul Ekman model of emotion (every possible emotion can be described by a weighted sum of six core emotions: happiness, sadness, anger, fear, disgust and surprise.) These weights are inputted into a decision making structure. The game writer defines emotions and the associated multiplicative weights of the six core emotions, using a language called Feelings Modelling Language (FML) (which is a subset of CML and can be externally referenced similarly to DTML - see Section 5.5). The NPC's mood is determined by selecting the emotion with the greatest output value based on the inputted core emotion weights. See Figure 5.8 for an example. This technique of determining the mood is extremely difficult to visualise, making creating the NPC difficult. As such it was decided to make use of a second textual affect sensing framework as well, making it the game writer's responsibility to choose which textual affect sensing algorithm they would like to use. SentiStrength outputs a positive and negative rating of text. These values are then summed and if the resulting value is positive then the NPC is positive, if the resulting value is negative then the NPC is negative and if the resulting value is zero then the NPC is neutral. It is the responsibility of the game writer to determine what these distinctions (positive,

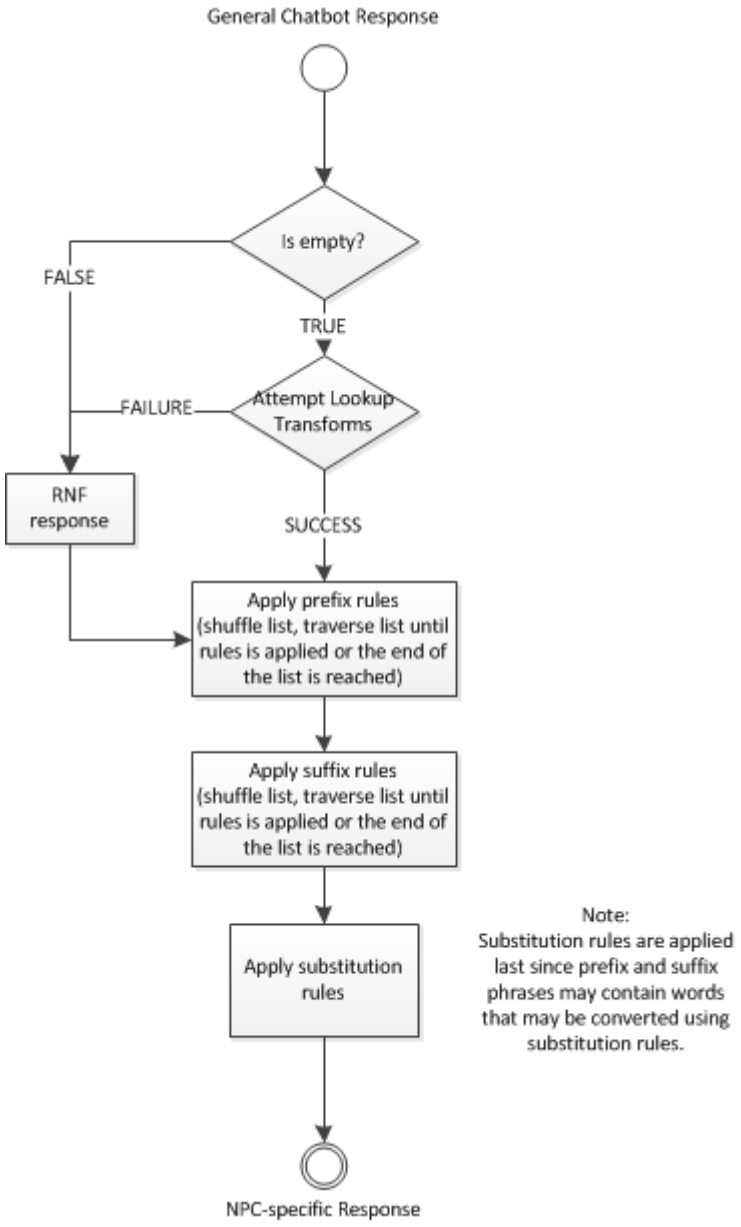

Figure 5.7: The translation process.

neutral and negative) imply.

5.7 NPC GUI

A GUI was created to simplify the process of creating NPCs and prevent game writers from having to write the CML code needed to model the game characters, thereby making the NPC creation process easier to understand and quicker. As well as the creation of CML the GUI allows the user to create various templates

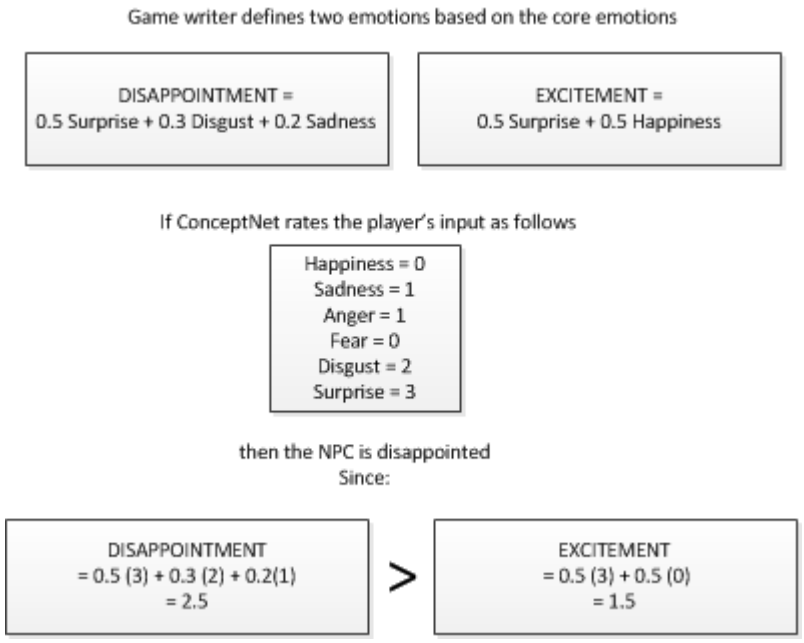

Figure 5.8: An example of the mood model using ConceptNet.

(LTML, DTML and FML) as well as import these templates into a CML document. The original design of the GUI (see Figure 5.9) featured a similar C# tree structure to that used in the general chatbot GUI (see Section 5.4), to visualise the CML document structures (namely lookups, dialect and mood). From feedback it was determined that this GUI was rather counter-intuitive and it was decided to redesign it so that it has the appearance of a spreadsheet that simply requires the game writer to fill in the necessary fields. The redesigned GUI (see Figure 5.10) has C# tables for the lookup, dialect and mood sections of the CML document. The game writer is able to select which textual affect sensing algorithm is used as well as specify RNF responses.

5.8 Conclusion

The prototype systems consists of markup files (which are used to store the knowledge of the general chatbot as well the NPC models), which are created using GUIs and are then uploaded onto a webserver which performs the processing. A game acts as client and communicates with the webserver using URL-encoded requests. The general chatbot was created using PyAIML, a Python-based AIML interpreter. The mood is modelled using two different approaches based on the ConceptNet (a Python-based framework) and SentiStrength (a Java-based framework). Dialect is modelled using the concept of emergence - three simple probabilistic rules (substitution, prefix and

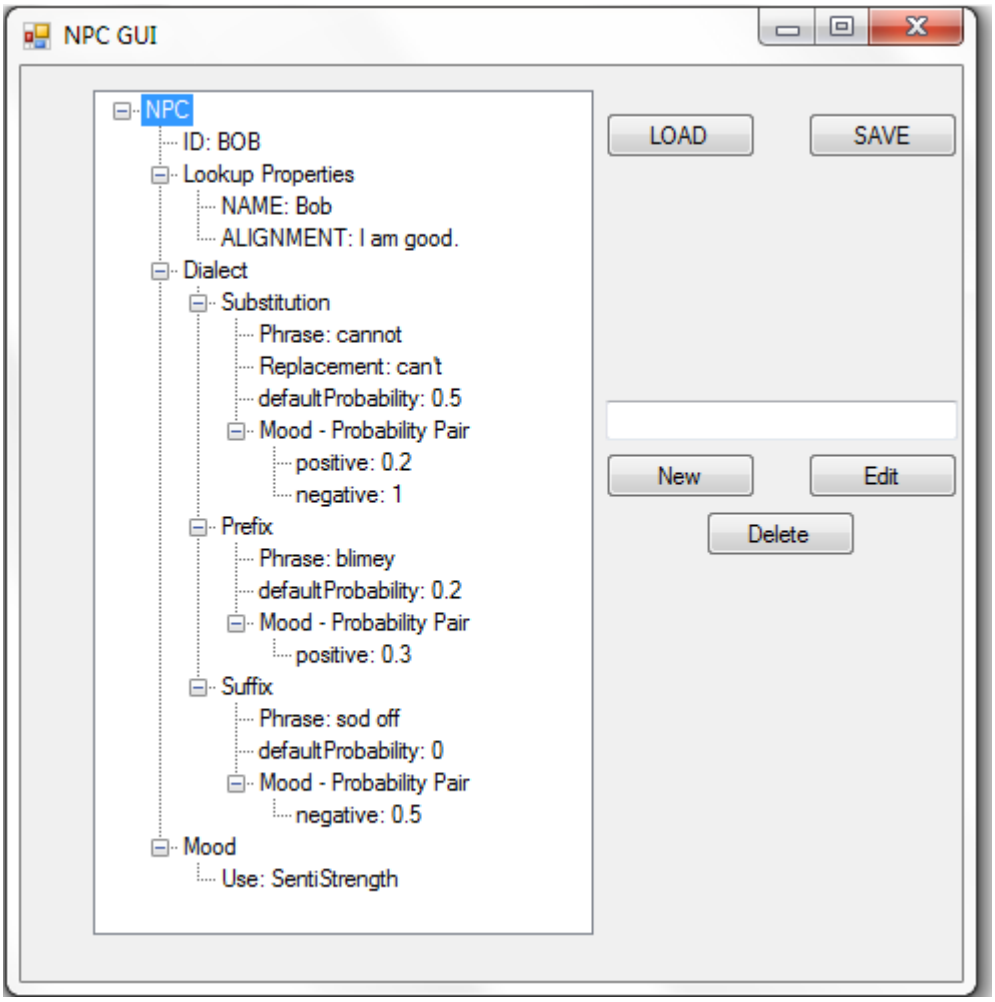

Figure 5.9: The original design of the GUI for creating NPCs.

suffix) to create unpredictable behaviour and the illusion of complexity.

This chapter contains the system implementation details including a high-level overview and the models of dialect and mood used. The following chapter describes the validation process including the generation of tests and the qualitative results obtained from these tests.

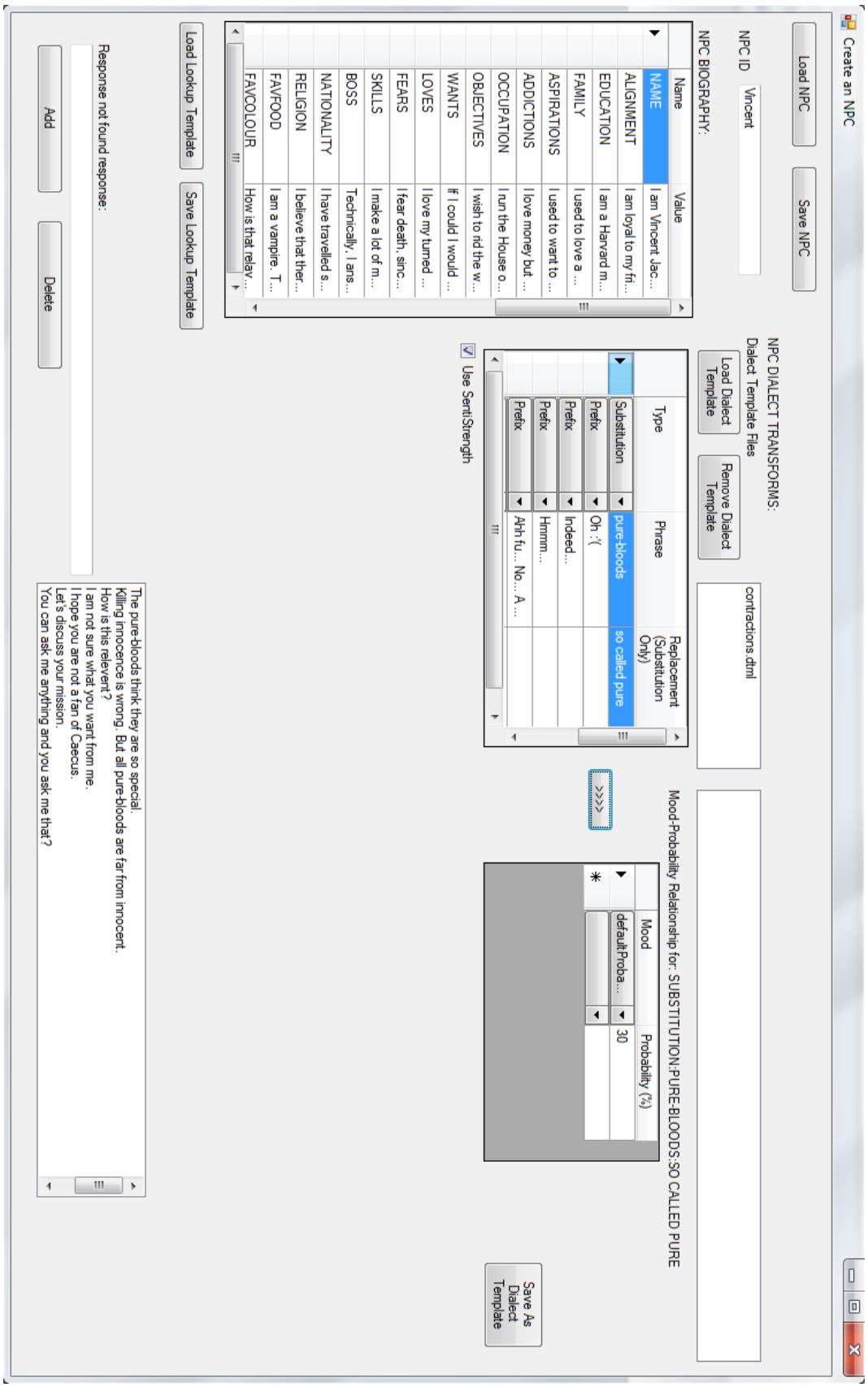

Figure 5.10: The final design of the GUI for creating NPCs.

Chapter 6

Testing and Results

This chapter presents the testing that has been performed. This includes a theoretical analysis of the variability of the transforms, practical testing and author testing. The practical testing was performed by creating a test game with metrics and have playtesters give feedback via an online questionnaire. Author testing allowed testers to interact with the authoring GUIs and fill out a questionnaire to gauge the hedonic qualities of the authoring tools. The generic responses of a single chatbot can be transformed to convey personas of different NPCs since the transforms are unpredictable (gives the illusion of intelligence), the test NPCs are effective at conveying information, the players noticed mood changes and accurately judge the personas.

6.1 Introduction

The purpose of testing is answer the research question, which is stated in Chapter 4 as:

Can the stock responses of a single chatbot be transformed to convey the personas of different NPCs and thereby reduce the amount of effort required to simulate conversation with different NPCs?

It was decided the research question can be answered by answering the following questions:

1. How variable are the transforms?
2. How effective are the NPCs at conveying information?

3. How well does this system allow the portrayal of different personas?
4. Did the player notice a change in mood?

A theoretical analysis (Section 6.2) can be used to answer question 1, since equations can be derived to show the number of unique NPC-specific responses generated from general stock responses. The number of unique responses can be a measure of unpredictability since a player will not be able to predict exactly what the NPC will say. This unpredictability makes the NPCs seem more intelligent and realistic [30]. The second question can be answered by seeing whether the players are able to get the NPCs' backstories and how often the NPCs do not understand player input. The rest of the questions depend on user experience and as such require a practical analysis. This has been done by creating a test game environment and allowing users to interact with the game. A qualitative analysis can be then performed using surveys (Section 6.3). Certain metrics (Section 6.3.5) can the be measured and interpreted to give a quantitative analysis of user experience. It must be stated that the practical analysis is not a true assessment of the system but rather an assessment of how well the tools have been used to create the NPC and as such, it is an indirect assessment of the system. Another focus of the research is providing a tool that is easy-to-use and understand, as well efficient to create NPCs, since existing Interactive Fiction systems with similar functionality require a deep technical understanding which most writers do not have or are tedious to use (see Chapter 3). The testing of these hedonic qualities can be seen in Section 6.4.

6.2 Theoretical Analysis

This section is a theoretical analysis of the transform rules and is included to show the variability of the NPC-specific responses that are procedurally generated from the general chatbot's stock responses given certain assumptions. As already mentioned variability makes the NPCs seem more intelligent and realistic [30]. For this section it is important to understand how the dialect transforms are applied, which is described in Chapter 5. The NPC mood affects the probability of transforms occuring and, as such, can be ignore in this analysis. Assuming a specific NPC has **p** prefix rules and **s** suffix rules then there are **p'** (Equation 6.1) possible prefix transforms and **s'** (Equation 6.2) possible suffix transforms.

$$p' = p + 1 \tag{6.1}$$

$$s' = s + 1 \tag{6.2}$$

This is because only one prefix rule can be applied at a time and only one suffix rule can be applied at a time and there exists the state when the transform does not occur. Since a prefix transform and a suffix transform can be applied simultaneously, one stock response can result in **a** (Equation 6.3) unique NPC-specific responses, assuming there are no substitution rules applied.

$$a = (p + 1)(s + 1) \tag{6.3}$$

Now assuming the stock response contains **r** different substitution phrases (ignoring the possible substitution phrases that may occur in the prefixes and suffixes) and each substitution has only one possible replacement, then each substitution is only a binary decision - it either happens or it does not. Since multiple substitutions can occur simultaneously the number of possible variations owing to substitution **r'** is like a binary number system with **r** digits, given by Equation 6.4.

$$r' = 2^r \tag{6.4}$$

So the number of unique NPC-specific responses for a stock response with the above mentioned assumptions is given by Equation 6.5.

$$n = (p + 1)(s + 1)(2^r) \tag{6.5}$$

Three graphs (see Figures 6.1, 6.2 and 6.3) have been created by varying different variables in Equation 6.5, to show how the number of unique NPC-specific responses scale. Figure 6.1 shows that there is a linear relationship between the number of prefix (or suffix) rules and the number of unique responses. Figure 6.2 shows the exponential relationship between number of substitution rules and number of unique responses. Figure 6.3 illustrates how drastically the number of unique responses increases if the number of each type of dialect rule simultaneously increase (hence the range of p = s = r = [0,10]).

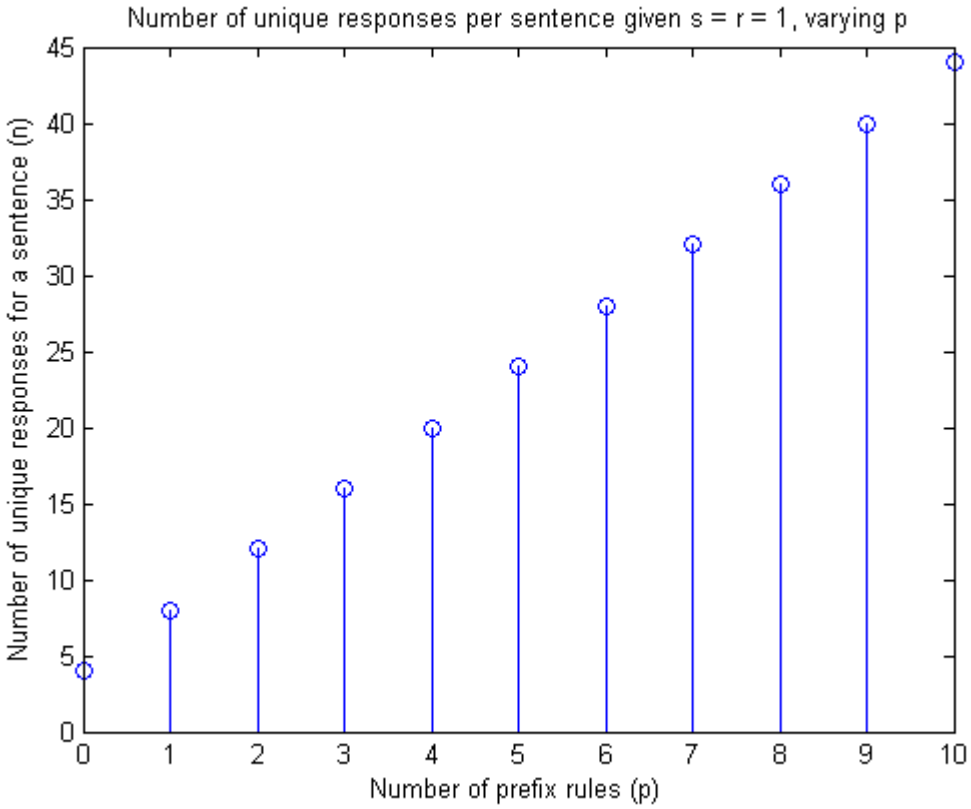

Figure 6.1: The number of unique NPC-specific responses for a stock response given s = r = 1 and varying p. The graph looks exactly the same for p = r = 1 and varying s.

In order to demonstrate how these transforms scale with a different number of general responses Equation 6.6 has been derived, by letting **i** denote the index of stock response in a list of stock responses. Each stock response may contain a different number of substitution phrases and therefore **r'** is different for each stock response. So if there are **k** general responses in the list then the total number of unique NPC-specific responses is:

$$n = (p+1)(s+1)\sum_{i=1}^{k} 2^{r_i} \tag{6.6}$$

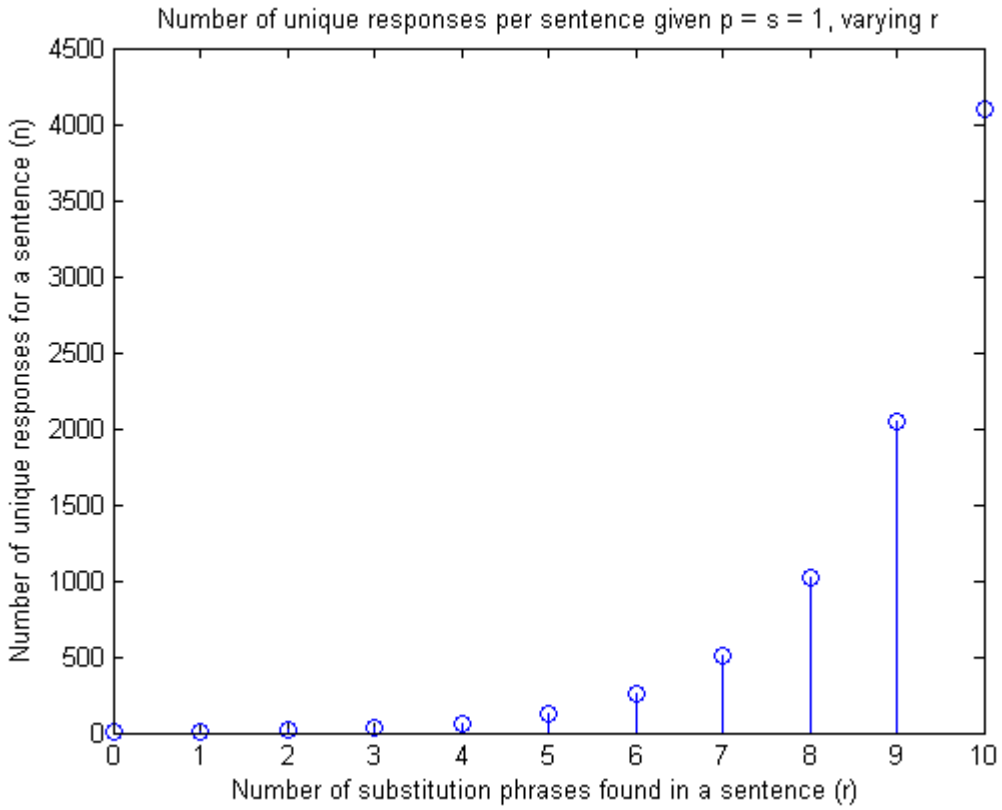

Figure 6.2: The number of unique NPC-specific responses for a stock response given p = s = 1 and varying r.

6.3 Interactive Testing

6.3.1 Test game implementation

The test game was created using Unity3D and is embedded in the webpage, so that users are able to test the game online. The player character is represented as a sphere and the NPCs are represented as cubes from a top-down perspective. This abstract representation was chosen so that the player cannot derive the NPCs' personas from their appearance. This ensures that the player derives the persona purely from the text responses of the NPC. The game world is populated with two randomly selected NPCs (Section 6.3.3). Upon starting the game the player is briefed with some instructions informing them of what to do and how to do it. They can walk up to an NPC which opens a chat window, allowing the player to interact with the NPC via typed natural language. A hint box is provided to give the player some limited guidance on what type about and cycles through the following hints (using

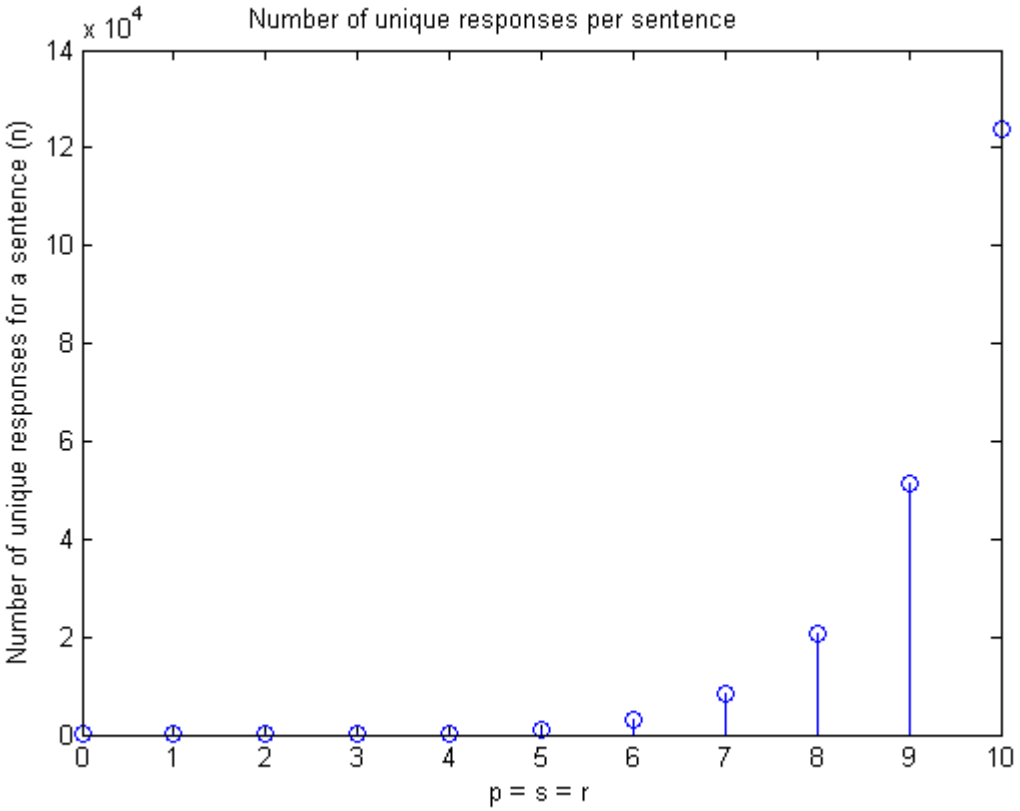

Figure 6.3: The number of unique NPC-specific responses for a stock response given p = s = r = [0,10].

the TAB key):

- Find out about your different missions.
- Where does this NPC's allegiance lie?
- Who does the NPC consider friends?
- Is this NPC educated?
- Who is this NPC's boss?
- What is the NPC's job?
- What scares this NPC?
- What does the NPC want to achieve?
- What does the NPC love?
- Does this NPC have any addictions?
- Remember that not all NPCs are created equal. Do not get frustrated. Rather

ask different questions.

- Does the NPC have a job?
- What must you do?
- Ask about someone the NPC has mentioned.
- Ask about a location that the NPC has mentioned.
- Rephrase questions that did not result in satisfying answers. They may yield different results.
- Ask the NPC about terms that he/she may have mentioned.
- Insult the NPC. See if you can change its mood.
- Compliment the NPC. See if you can change its mood.
- Ask game related questions.
- Try find out more about this character's backstory by asking relevant questions.
- What about this character do you want to know?
- Different characters have different levels of training. Do not get frustrated if a decent response is not given then ask a different question.
- Think about this as a character in a game, not a real-life character.
- Imagine you could phrase the questions in Mass Effect. What would you ask a game character?
- What is the NPC's name?

Once the player is done chatting to the first NPC they can click on a button, which opens a Google Survey (in a frame on the same page). The user can no longer chat to that NPC but can view the conversation (this is to prevent them from interacting with the NPC further and thereby skewing the results of the survey). The survey then states that the user must interact with the second NPC and then fill out questions about mood. For more information on the survey (also referred to as a questionnaire in this dissertation) see Section 6.3.4.

6.3.2 General Chatbot

The majority of the authoring time was spent on creating the general chatbot since this acts as the knowledge-base for all of the NPCs. Since the test environment is a game the focus was to create AIML patterns (using the Authoring GUI) that relate to the game story and characters and as such the player will get a "Response not found" response if they type something not related to the game. Some initial testing

was done to find as many synonyms for the game-related patterns as possible. Some game related patterns and synonyms can be seen in a screenshot of the General Chatbot authoring tool (Figure 6.4). The general chatbot makes use of several lookup transforms that are tags that converted to NPC-specific answers. NPCs have different biographical details (such as name, age, wants etc) and world views (of other NPCs and the environment), which are represented by these lookup transforms. The NPCs' relationships and view of the environment were very carefully planned. It can be seen in Figure 6.5 that creating the general chatbot took the most time. The general chatbot is minimal trained as only 3 days were spent creating the AIML files which add up to 120 kB of hard drive space. Compare this to A.L.I.C.E, which has been in development for 18 years and the AIML files take up over 10 MB of hard disk space [6], and one can see that the general chatbot is minimally trained.

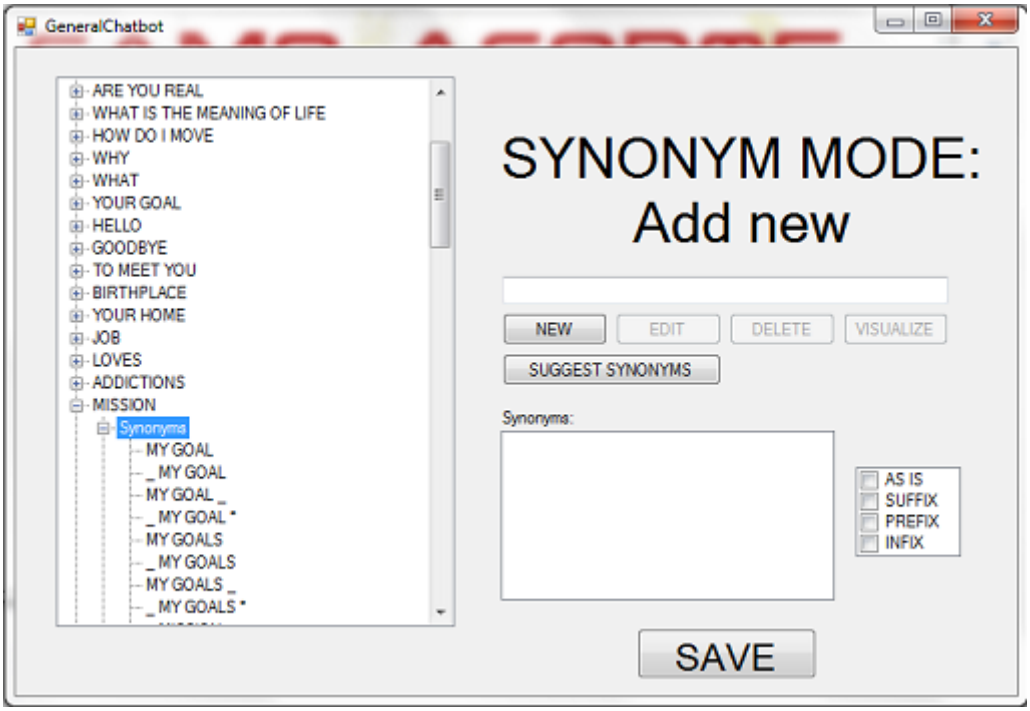

Figure 6.4: A screenshot of the General Chatbot authoring tool which shows some of the game related patterns used in authoring the general chatbot. The MISSIONS pattern synonym list has been expanded to show some of the synonyms.

6.3.3 NPCs

This section assumes the reader is familiar with the implementation of the authoring tools and the translation layer (see Chapter 5). The game is about different groups of vampires, called Houses. The setting is quite dark, with betrayal and murder.

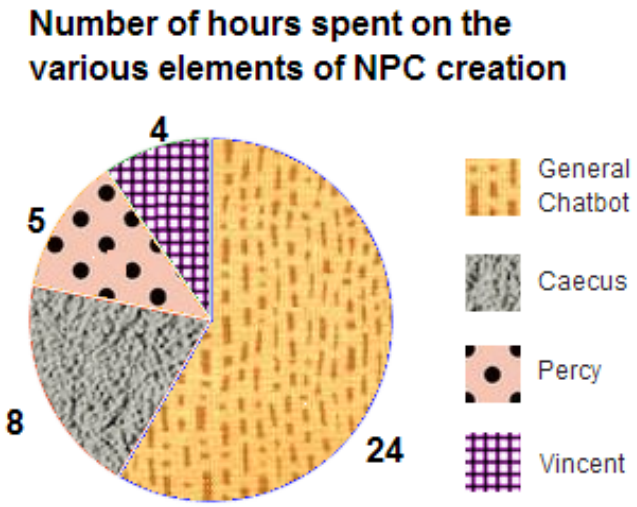

Figure 6.5: Time spent on different elements of NPC creation.

Each NPC was created using a short description (showed in bold font below) which summarises the character. The three NPCs are vastly different, allowing it to be assessed whether the NPCs are correctly portrayed to the player and assess whether they convey information effectively. The survey and the NPCs were created iteratively since one is dependant on the other. For example, the survey contains a list of personality traits (based on [2] and [36] and modified to take nationality into account), NPCs were created using this list and once some testing was performed extra traits were added to the list. The final personality trait list can be seen in Table 6.2 in Section 6.3.4. The authoring of the NPCs consisted of creating the general chatbot and creating the NPC-specific transforms. As mentioned in Section 6.3.2 the NPCs make use of lookup transforms. The NPCs have different levels of training resulting in characters with different amount of backstory (hereon referred to as developed, partial and undeveloped backstories). This was done using LTML documents as described in Chapter 5. Dille et al [2] provides a list of biographical characteristics, and an example of a well-rounded player character is given. The biographical characteristics list was converted into the different lookup transforms. Characters with developed backstories were given all of the transforms while characters with partial backstories have less lookup transforms and characters with undeveloped backstories have the least. There are other characters purely for the test game's story. Each NPC has knowledge about these characters, as well as the environment. This was taken into account by adding lookup transforms. The complete list of lookup transforms for developed, partial and undeveloped backstories can be seen in Table 6.1. All of the NPCs make use of SentiStrength for textual

affect sensing, as using ConceptNet proved difficult to get accurate mood changes. The NPCs have specific responses when they cannot understand the player's input. These are called "Response Not Found" responses. These have been designed in such a way to try convey the NPCs' personas and also guide to the player's conversation.

Table 6.1: The different LTML documents contain the lookup transforms as indicated by X.

Lookup Transform	Description	Undeveloped	Partial	Developed
$NAME	The NPC's name	X	X	X
$ALIGNMENT	Who are the NPC's allies?	X	X	X
$EDUCATION	How/where was the NPC educated?			X
$FAMILY	Information about the NPC's family.		X	X
$ASPIRATIONS	What does the NPC aspire too?			X
$ADDICTIONS	Does the NPC have any addictions?			X
$OCCUPATION	What is the NPC's job?	X	X	X
$OBJECTIVES	What does the NPC hope to achieve?	X	X	X
$WANTS	What are the NPC's wants?		X	X
$LOVES	What does the NPC love?			X
$FEARS	What scares the NPC?	X	X	X
$SKILLS	What skills does the NPC possess?	X	X	X
$BOSS	Who does the NPC work for?	X	X	X
$NATIONALITY	What is the NPC's nationality?		X	X
$RELIGION	What are the NPC's religious beliefs?			X
$FAVFOOD	What is the NPC's favourite food?			X

Table 6.1: The different LTML documents contain the lookup transforms as indicated by X.

Lookup Transform	Description	Undeveloped	Partial	Developed
$FAVCOLOUR	What is the NPC's favourite colour?			X
$BIRTHPLACE	Where was the NPC born?		X	X
$HOME	Where does the NPC live?		X	X
$MORALITY	What morality does the NPC subscribe too?			X
$COMFORTS	What comforts the NPC?			X
$TAGLINE	This is what the NPC is famous for saying.	X	X	X
$AGE	The NPC's age.	X	X	X
$MISSION1	One of the missions that the NPC can give the player.	X	X	X
$MISSION2	One of the missions that the NPC can give the player.	X	X	X
$CAECUS	What does the NPC think of Caecus?	X	X	X
$VINCENT	What does the NPC think of Vincent?	X	X	X
$PERCY	What does the NPC think of Percy?	X	X	X
$GABRIEL	What does the NPC think of Gabriel?	X	X	X
$SAMANTHA	What does the NPC think of Samantha?	X	X	X
$DIETRICH	What does the NPC think of Dietrich?	X	X	X
$MELISSA	What does the NPC think of Melissa?	X	X	X

Table 6.1: The different LTML documents contain the lookup transforms as indicated by X.

Lookup Transform	Description	Undeveloped	Partial	Developed
$ALEXANDER	What does the NPC think of Alexander?	X	X	X
$HUNTER	What does the NPC think of Hunter?	X	X	X
$STEINER	What does the NPC think of House of Steiner?	X	X	X
$EREBUS	What does the NPC think of House of Erebus?	X	X	X
$JACKSON	What does the NPC think of House of Jackson?	X	X	X
$VAMPIRE	Is the NPC a vampire?	X	X	X

Vincent

Vincent was a successful lawyer when the vampires attacked, turning him and killing everyone he ever loved. Now he uses his business acuity to run a shelter for newly-turned vampires.

Vincent has a developed backstory (in terms of the LTML document used) and as such uses all of the lookup transforms (see Table 6.1). Even though Vincent has the most developed backstory he took the least amount of time to create out of all of the game characters (see Figure 6.5). This seems counter-intuitive but from the above description it can be seen that nothing is mentioned about dialect and mood and as such less effort had to be made when creating the dialect transforms. This is because the neutral responses of the general chatbot required only minimal tweaks to display the neutral character of Vincent. Vincent makes use of a template of dialect transforms containing common contractions. The "Response Not Found" responses were created to reflect Vincent's compassion (e.g. "The House of Jackson looks after everyone innocent.") and also references other characters (e.g. "Did Alexander send you?"), to encourage the player to ask questions about the other characters.

Caecus

Caecus is a cruel, arrogant, pure-blooded vampire from Eastern Europe (Czech Republic or some place similar) and she displays prejudice to any turned vampire.

Caecus has a partial backstory (see Table 6.1). It took the longest amount of time to create Caecus because her dialect needs to reflect that she is from Eastern Europe. This was achieved by translating some of her words into Czech and substituting the "W" with "V" to convey that stereotypical Transylvanian vampire pronounciation (e.g. I vant to suck your blood). Caecus does not make use of any dialect templates. The "Response Not Found" messages were created to illustrate Caecus' arrogance (e.g. "How can you talk to a pure-blood like that?").

Percy

Percy is an alcoholic, British hooligan who is working as Caecus' slave in return for the promise of being immortalised via a vampire bite.

Percy has an undeveloped backstory (see Table 6.1). His alcoholic nature is clearly reflected with prefix and suffix transforms of onomatopoeia indicating hiccups from exessive consumption. He is also conveyed as an alcoholic through his "Response Not Found" responses - six out nine of the responses are references to getting drunk. For Percy it was very important to convey him as British hooligan so that his dialect is vastly different to Vincent's dialect. This was achieved by incorporating many British expressions (such as "Guv" and "innit") found in British media, such as websites [37] [38] and British comedies (such as Fawlty Towers and Blackadder). His hooliganism is displayed through rude British expressions (such as "sod off" and "shite"), also found in British media.

6.3.4 Questionnaire

The questionnaire was created using Google Forms, which makes it easily accessible online and provides tools for analysis. The player experience questionnaire contains the following sections:

1. Personal Details - non-invasive questions to gauge experience (gaming, game development) in order to contextualise the responses. Players are able to fillout

the survey more than once and this section also has a question to determine how many times the player has filled out the survey.

2. NPC Experience - questions to assess what information the player was able to gain from the NPC and to see if the the character's persona is conveyed.
3. Mood Experience - questions to see if the player is able to detect mood changes in the second NPC.
4. General Experience - questions to assess what the player's overall opinion of the system is.

In NPC Experience Section the player is asked to choose which character they interacted with. The player is then asked to fill in information about the character (such as age, nationality etc). This is to assess the effectiveness of conveying information. The player is then given a set of traits and asked to select the ones that apply to the NPC. This is to test the effectiveness of portraying a persona as the player answers can be compared to traits defined during the authoring process. The list of traits in the survey and the author's assessment of each character (based what the author was hoping to portray) is given in Table 6.2. The table contains some unused traits because those were presented to the playtesters to add noise to the selection. In the Mood Experience Section, the player is asked whether the NPC's mood changed, how often and how the NPC responded to different treatment (compliments versus insults). This is to assess the effectiveness of conveying mood. In the General Experience Section the player is asked to rate the following aspects of the NPC, and the system, using the Likert Scale (strongly disagree, disagree, neutral, agree, strongly agree) [39]:

- Enjoyable: the purpose of gaming is to be fun and as such the interacting with the NPC via text has to be fun for the system to be viable.
- Frustrating: this influences the effectiveness of conveying information (see Section 6.3.6).
- Text-based: this acts as a control question, since the system is clearly text-based.
- Informative: this influences the effectiveness of conveying information (see Section 6.3.6).
- Realistic: this influences the effectiveness of conveying information and the effectiveness of portraying a persona (see Section 6.3.6).
- Entertaining: this is to see if the players found the system to be entertaining. This is because some people may find something entertaining without finding

it enjoyable and vice versa. Games need to be enjoyable and/or entertaining to the player.

- Preferable to cut-scenes: to assess whether the players enjoy the agency provided.
- Unpredictable: as mentioned, unpredictability gives the illusion of intelligence and realism.
- Revolutionary: to assess whether the player feels that the system could revolutionise gaming.
- Has lots of potential: with extra work could the system bring something new and special to the gaming industry.
- Disappointing: to assess whether the player feels that the system could have been done better.
- Pointless: does the player feel that the system can make no contribution to gaming.

Table 6.2: The different personality traits that each character has. This list was presented to playtesters to rate the NPCs and as such some unnecessary traits were added to add noise to the selection.

Trait	Vincent	Caecus	Percy
Classy	X		
Alcoholic			X
Humble			
Rude		X	X
Cruel		X	
Kind	X		
Humanitarian	X		
Arrogant		X	
Hooligan			X
Eastern European/East German		X	
British			X
Trustworthy			
Snob		X	
Ignorant			X
Delusions of Granduer			
Vengeful	X		
Stable	X		

Table 6.2: The different personality traits that each character has. This list was presented to playtesters to rate the NPCs and as such some unnecessary traits were added to add noise to the selection.

Trait	Vincent	Caecus	Percy
Volatile		X	X
Devoted	X		
Assertive			
Aggressive			
Good-natured			
Lazy			X
Organised			
Business Acuity	X		
Patient			
Impatient		X	
American			
Friendly			
Loyal			
Selfish			
South African			
Depressed			
Carribean			
Drug addict			

6.3.5 Metrics

Metrics are measurements made by the game automatically without the player's knowledge [40]. These measurements are made so that developers can gauge players' styles and update the game experience to suit the players [41]. Metrics have been implemented in the test game in the form of log files, which can be analyzed using a Python script. The log files are created by the game by storing the conversation between the player and the NPC and by logging the following events:

- Stimulus-Response Pairs (SRPs): each interaction between the player and NPC (player's message and NPC's response) is counted.
- "Response Not Found" responses (RNFs) are tagged. The number of occurences of this tagged can be compared to the total conversation length, as measurement of the effectiveness of the NPC of conveying information (i.e. the ratio of RNFs

to actual responses).

- Mood changes are tagged and can be used to indicate how often the mood changed, on average, per conversation.

These values are averaged over the total number of conversations and the average conversation length is also measured. See Section 6.3.6 for the results. See Appendix A for an excerpt of a conversation.

6.3.6 Results

User Experience Testing

The average conversation length is approximately 22.1 SRPs. The average number of RNFs is approximately 7.4. This implies that approximately 33.6% of the time the NPCs did not understand the user input. Considering that the general chatbot is minimally trained (see Section 6.3.2) and that the different NPCs have different levels of training, this shows that the NPCs are good at conveying information since the RNFs are in the minority. There are an average of four mood changes per conversation, implying that every 5.5 SRPs the NPCs mood changes. This shows that the NPCs are able to change their moods. There were 42 responses to the questionnaire. 7 responses were removed because of an incorrectly answered control question. The remaining responses came from the following demographic:

- Most of the responses were from participants that have jobs (31%) or are studying at an undergraduate level (34%).
- Most of the responses were from gamers: 49% of the players stated that they often play games, while 23% stated that they are avid gamers and spend most of their free-time playing games. 23% stated that they seldom play games.
- Most of the players have never developed a game: 40% stated that they have never made or designed a game and 14% stated that they have designed games but never actually made one.
- 80% of the responses are from players doing the questionnaire for the first time.

Of the 35 responses, 42.86% came from the players that interacted with Caecus and 28.57% responses came from players that interacted with Vincent. The rest of the responses (28.57%) came from players that interacted with Percy. The percentage of acceptable answers in terms of biographical information about the NPC is given

in Table 6.3. A response is rated as acceptable if it matches the NPC's backstory, makes sense according to how the NPC dialect and if no response is given where the NPC does not have any backstory elements related to the question. As can be seen in Table 6.3, the average percentage of acceptable answers are 52.5%, 43.33% and 70.83% for Vincent, Caecus and Percy. These results may seem counter-intuitive since Percy has an undeveloped backstory but has a higher average. This could be attributed to interaction with smarter players, his responses may guide the player better but the more likely reason is the fact that less information is required for an acceptable answer because he has less of a backstory. The weighted average percentage of acceptable responses (calculated by taking the number of players which interacted with each NPC into account) is 53.81%. This value is high considering that the players had limited guidance (in the form of a hint box) on what questions to ask and since they had the freedom to ask anything. As already mentioned, 80% of the player's were doing it for the first time, so they would struggle to know what to ask. Another factor to consider is the limited amount of time that players had. Taking all of this into account, the NPCs can be considered to be good at conveying information to the player. Another thing to consider is the fact that the practical testing does not test the overall system but rather how well the tools were used.

Table 6.3: The percentage of acceptable answers to the biographical questions regarding each character.

Character Information	Vincent	Caecus	Percy
Age	30	13.33	10
Alignment/Friends/Loyalty	70	60	90
Education	30	80	60
Family	80	33.33	100
Favourites (food and colour)	20	66.67	100
Fears	60	33.33	100
Goals and Aspirations	80	40	70
Birthplace and Nationality	10	33.33	30
Employment, Skills and Boss	80	53.33	50
Beliefs, Morality and Comforts	50	33.33	90
Loves, Wants and Addictions	50	40	80
Player's Missions	70	33.33	70
Average	52.5	43.33	70.83

The respondents identified which personality traits (see Table 6.2) they felt the NPC that they had interacted with had. The number of respondents to check each

trait for each NPC was counted and calculated as a percentage of the number of respondents to interact with the NPC. An NPC is considered to have a certain trait if the percentage is 50% or more, since it was decided to treat this as a majority vote and these traits are added to the "majority vote" set. This was done so that"majority vote" set can be compared to the traits that the author planned ("author" set) for each NPC (see Table 6.2).

1. The traits that occur in both the "majority vote" and the "author" set are added to a set called "common". This is effectively an AND operation.
2. The traits that occur in either "majority vote" or "author" set or both are added to a set called "total". This is effectively an OR operation.
3. The number of traits in the "common" set is the calculated as a percentage of the number of traits in the "total" set and referred to as "persona accuracy" below.

Vincent has the lowest persona accuracy of 42.86%, followed by Caecus at 50% and Percy at 66.67%. The weighted average (calculated by taking the number of respondents per NPC into account) is 52.72%. Percy was the best at conveying his persona, with a "persona accuracy" of 66.67%. The reason that Percy has a higher persona accuracy than Vincent is because Vincent is a fairly neutral character while Percy is more obvious (see Section 6.3.3). As such the system allows authors to portray different personas well.

System Experience Testing

The Likert Scale ratings made by the players with respect to the NPC that they interacted with can be seen in Figure 6.6. The qualitative statements that follow are based on this figure and compare the number of positive (agree and strongly agree) responses to the number of negative (disagree and strongly disagree) statements. Consult the figure for specific values. Most of the players found the game enjoyable despite the fact that they found it frustrating. These respondents all answered the control question favourably (agreed that the interaction is text-based). The feedback on whether the interaction was informative and preferable to cut-scenes was neutral, tending towards slightly positive. A large majority of the respondents found the NPC to be unrealistic, but still found it entertaining. The NPCs are considered unpredictable by most of the respondents. The fact that the majority of the respondents find the system unrealistic negatively effects the effectiveness of conveying information and effectiveness of portraying a persona ratings, however,

the majority of the people found the game entertaining and enjoyable. Since this is the goal with gaming these negative attributes can be offset and it can be stated that the NPCs are effective enough at conveying information. The fact that most people found interacting frustrating can be attributed to the lack of guidance and the fact that players can literally type anything to the NPCs, resulting in questions that do not have answers. This implies that developers using the system have to balance the amount of guidance they give the players and the verbosity of the NPCs (models and the general chatbot knowledge-base). Interacting with the characters therefore becomes a skill that the players must learn and as such can offer a rewarding experience.

The system experience ratings based on the Likert Scale can be seen in Figure 6.7, consult the figure for specific values as the following discussion is purely qualitative. Respondents offered neutral feedback (only slightly positive) about whether the system is revolutionary or not. The majority of respondents did, however, indicate that the system has lot of potential and as such further development of the system could possibly revolutionise gaming. The minority of the respondents found the system disappointing or pointless. These four ratings indicate that the research is indeed viable and necessary.

Mood Experience Testing

Mood changes were assessed via a questionnaire (see Section 6.3.4). Most of the respondents were able to determine the NPC's initial mood, with only 14% stating that the mood was indeterminable. Only 29% of the respondents did not notice the mood changes, while 17% said it was not very obvious, 6% said maybe, 26% said the mood change was not realistic and 17% stated that there was a definite change in mood. An equal amount of respondents classified the NPC as Volatile and Very Stable, at 23%. When asked how the NPC responded to negative treatment, only 7% of the respondents stated that there was no noticeable change in mood. This figure is a lot higher (40%) for positive treatment. While the results show that most players were able to see the NPC's mood, this element of the system needs improvement (see Future Research).

Summary of Results

The summary of the results can be found in Table 6.4.

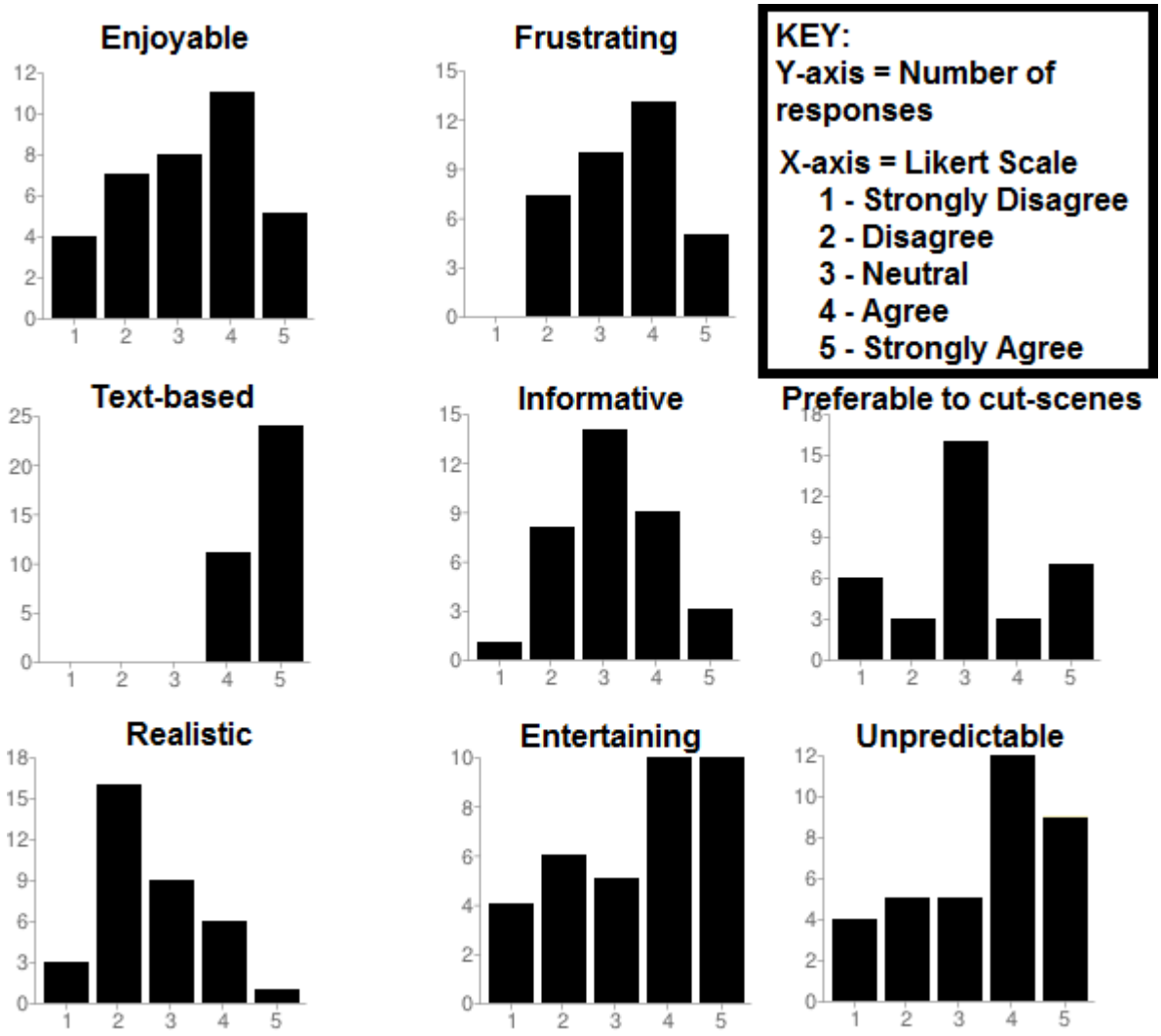

Figure 6.6: Graphs indicating how the players rated the NPC according to specific criteria, using the Likert Scale

Table 6.4: Summary of the results.

Question	Qualitative Answers	Motivation
How variable are the transforms?	Extremely variable - this is desirable.	Theoretical Analysis, Unpredictability Likert Scale rating.

Table 6.4: Summary of the results.

Question	Qualitative Answers	Motivation
How effective are the NPCs at conveying information?	Good enough at conveying information.	NPC with most correct biographical answers = 70.83%. Average correct biographical answers = 53.81%. Percentage RNFs of SRPs = 33.6%. Negatively influenced by player frustration and lack of realism.
How well does this system allow the portrayal of different personas?	Can portray personas well, depending on how well the character has been written.	Maximum persona accuracy = 66.67%. Average person accuracy = 52.72%.
Did the player notice a change in mood?	Yes, however, the mood changes are harder to convey.	Average of four mood changes per conversation. 29% of respondents stated there was no mood change and 23% classified the NPCs as very stable.
At an implementation level: "Can the stock responses of a single chatbot be transformed to convey the personas of different NPCs and thereby reduce the amount of effort required to simulate conversation with different NPCs?"	Yes	The results mentioned above indicate this.

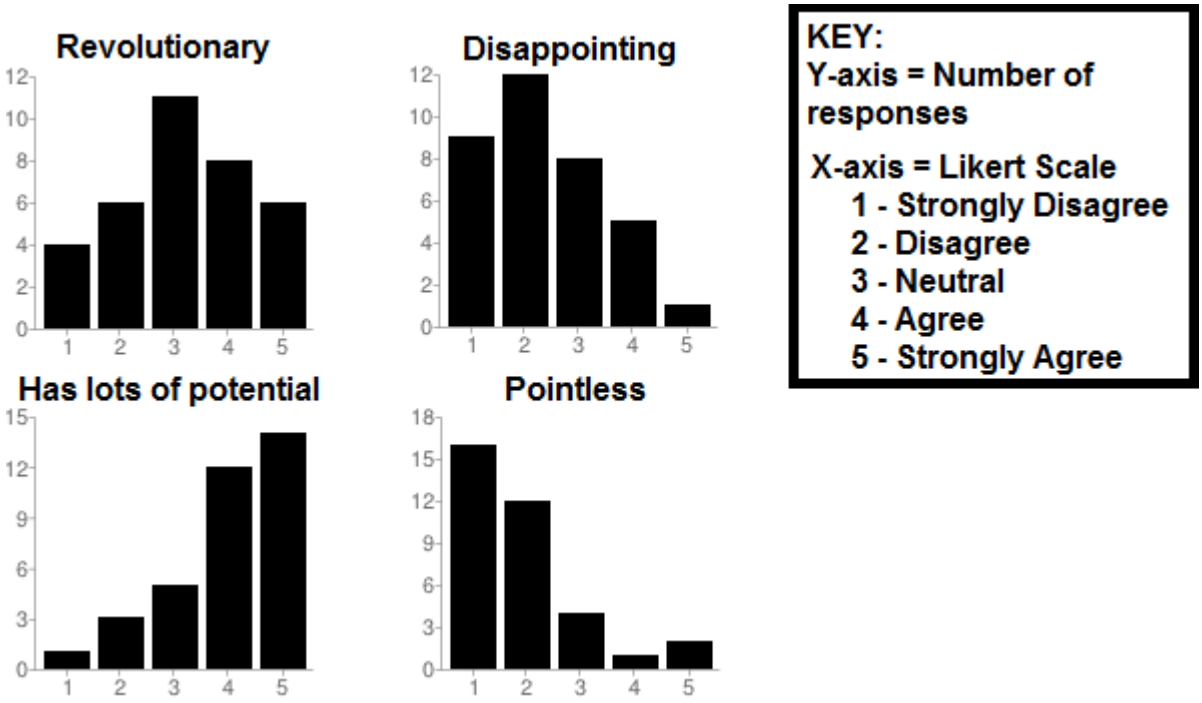

Figure 6.7: Graphs indicating how the players rated the system according to specific criteria, using the Likert Scale

6.4 Authoring Tool Testing

6.4.1 Questionnaire

As can be seen by Figure 6.5, NPCs take many hours to create. As such it was impractical to get each user create a character (since it was difficult enough to get people to interact with the test game and fill out a short questionnaire). As such, each user was given limited instruction (approximately ten minutes each) by using the NPCs and general chatbot created for the interactive testing as examples. Each user was then allowed to interact with the GUIs and test the functionality it provides. The users were then ask to fill out a questionnaire, on the computer. The questionnaire starts with Likert Scale questions, to get ratings (easy to use, tedious, efficient, fun to use, feature-rich and understandable). The users were asked open-ended questions about what they liked best about the system and what disliked the most about the GUI. This is to ascertain if there are any obvious problems with the GUI and to see the strengths and weaknesses of the GUI. The user is then given the opportunity to comment on the GUI.

6.4.2 Results

Five users with different educational backgrounds were selected. This was to gauge the different levels of understanding and compare them for technical and non-technically minded people. The group comprised of a qualified electrical engineer, a qualified writer, an indie game developer, a game design student (majoring in writing) and a lay-person. All the users had relatively similar answers with respect to the Likert Scale questions, except the lay-person, who found the software a bit more difficult to understand. The average value of each Likert Scale rating was calculated (overall average) and compared to the average of ratings excluding the lay-person's responses (exclusive average). These averages are given in Table 6.5. As can be seen the authoring tool is easy to use and understand, efficient and not tedious, fun to use and feature-rich. Two out of the five users thought the general chatbot visualiser was the best feature, while the remaining three preferred the templates (lookup and dialect templates). Everyone found the authoring of the general chatbot a bit difficult and stated that this could be better visualised. Another concern was brought up that a decent understanding of the field is needed to use the software. Overall, the system is easy to use and efficient and, as such, is a success. Some future improvements could be made to the general chatbot authoring GUI. The users made the following suggestions:

- Visual programming of the general chatbot.
- Add groups to the general chatbot so that certain NPCs can have access to certain information, while others do not.
- List synonyms alphabetically.
- Visualise the synonym properties (as is, prefix, suffix and infix) instead of listing them separately.
- Have a testing environment built into the authoring tool, with some basic debugging capabilities.

Table 6.5: Results of the Likert Scale questions (1 - strongly disagree, 5 - strongly agree)

Characteristic	Overall Average	Excluding Layperson
Easy to use	4.2	4.5
Tedious	1.6	1.5
Efficient	4	4.25
Fun to use	4	4

Table 6.5: Results of the Likert Scale questions (1 - strongly disagree, 5 - strongly agree)

Characteristic	Overall Average	Excluding Layperson
Feature-rich	4.6	5
Understandable	3.8	4.25

6.5 Conclusion

The generic responses of a single chatbot can be transformed (using dialect and mood) to convey the personas of different NPCs. This has been proved using theoretical and practical analysis. The transforms are extremely variable (theoretical analysis and unpredictability Likert Scale rating), giving the illusion of intellect and realism. The NPCs are acceptable at conveying information. Most players acknowledged the mood change but felt that it should be less subtle. The NPCs' personas can be perceived in what they say. All these goals are achieveable through the use of an easy to use, understandable and efficient authoring tool. The general chatbot tool could be improved, to better visualise AIML so that it is more understandable.

The following chapter contains the conclusions drawn, a summary of the contributions of the author and suggestions for further research.

Chapter 7

Conclusion

7.1 Conclusions

Creating NPCs which respond to text falls into three fields of research: game development, interactive fiction and textual affect sensing. The prototype NPC system and authoring tools improve on the shortcomings of existing IF systems such as Scenejo [27], Prom Week [26] and Façade [25]. The proposed system is easy-to-use for non-technical people as opposed to the difficulties presented by the ABL programming language used by Façade. The proposed system makes use of the code re-use paradigm by allowing the user to create one AIML knowledge-base and create translation layers to model each NPC. This makes it scalable, unlike Scenejo which requires a knowledge-base per character per scene. Prom Week and Façade suffer from set, predictable dialogue, while the proposed system's transforms are probabalistic in nature and as such the dialogue is emergent. The research question based on the shortcomings of these systems is:

Can the stock responses of a single chatbot be transformed to convey the personas of different NPCs and thereby reduce the amount of effort required to simulate conversation with different NPCs?

The answer was determined through theoretical and practical analysis. The theoretical analysis was performed by deriving equations to show the variability of the transforms. Practical analysis was performed by creating a test game, with three vastly different NPCs, and having users interact with the game. Results were obtained through metrics and through voluntary surveys. The research question was answered by breaking it into four questions, with qualitative answers and motivations based on

the surveys and metrics. Firstly, the transforms are extremely variable, based on theoretical analysis as well as the fact that most testers found the NPCs' dialogue unpredictable. Secondly, the NPCs are effective at conveying information as they were able to respond to most of the players' inputs (only 33.6% of the NPCs responses were "Response Not Found" responses) and the players were able to get most of the NPCs' biographical information (53.81% on average, with a maximum of 70.83%). Thirdly, the NPCs' personas can be portrayed through their transformed text (players chose the traits they believed that the NPCs had and a maximum accuracy of 66.67% was achieved and the average accuracy is 52.72%). Lastly, the NPCs' moods change, but should be more obvious (average of four mood changes per conversation and only 29% of the players did not notice a mood change). From the above it can be seen that the stock responses of a single chatbot can be transformed to convey different NPC personas but is best suited to personas that are descriptive as opposed to more neutral characters (see Percy and Vincent respectively in Chapter 6). The testing of the hedonic qualities of the software revealed that most people found the authoring tool ease to understand and use, but the GUI for creating the general chatbot could be simplified.

7.2 Summary of Contributions

The focus of the research is on creating a translation layer that can convert the generic responses of a single chatbot to portray different personas of different NPCs. The transform layer required the development of a model of dialect. The proposed model contains three types of rules that are applied to a generic response, namely substitution, appending to the front (prefix) and appending to the back (suffix). These rules are defined by the author, per NPC, who gives them a probability of occurrence based on the NPC's mood. This model and the translation layer is a novel approach to create NPCs that the player can talk to using typed natural language. It allows for a single chatbot to be used instead of having a chatbot per NPC. This makes the system scalable and saves time as training a chatbot takes a lot more time than defining the NPCs' dialects (as can be seen in Figure 6.5, Chapter 6). The author defined a modelling language to store the NPC models and the various template languages needed to streamline the creation process. Authoring tools, for creation of the general chatbot's AIML and creation of the NPCs' dialects, were also created and while users found them easy to understand and use they could be improved. A webserver was created to parse the various files and process the players' input. A test game was created to test the system and playtesters gave feedback

of their experiences. The focus of the research is not on creating a new chatbot architecture and as such AIML is used to store the general chatbot. The focus was not creating a new AIML interpreter and as such PyAIML [32] was used. The goal was not to create some novel textual affect sensing technique and so ConceptNet [22] and SentiStrength [16] frameworks were used. The focus of the research is on procedural dialogue generation and not plot management, and as such no AI system was created to control the story.

7.3 Recommendations for Future Research

During testing of the authoring tools it was discovered that users found training of the general chatbot a bit difficult to understand. This implies the need for some research into the visualization of AIML. This research could include visualization of all the functionality provided by AIML, instead of just the subset needed in this application. The existing A.L.I.C.E. AIML files could be used to test this visualization. Further research could be conducted into improving the GUIs, possibly by creating a visual programming interface such as Unreal Kismet [42]. Further research could be done regarding the dialect rules: some extra functionality could be added such as conditional "Response Not Found" responses, to provide better context. An AI system for managing the story, such as a drama manager, could be implemented and integrated with this system. This could provide collaboration between the author and creators of existing systems such as Prom Week [26] and Façade [25]. Further research could be conducted into improving the general chatbot creation process by using AI techniques to teach the general chatbot by communicating with it via typing. The existing text-based system could be integrated with speech synthesis and recognition software to create a truly immersive experience, where the player chats with a NPC using his/her voice and is able to hear the NPC's response.

References

[1] Rogers, S. (2010), *Level Up!: The Guide to Great Video Game Design*, Wiley.

[2] Dille, F. & Platten, J.Z. (2007), *The Ultimate Guide to Video Game Writing and Design*, Skip Press, USA, first edn.

[3] Video Game Sales Wiki (2012), 'Video game industry', http://vgsales.wikia.com/wiki/Video_game_industry. Last accessed 1 December 2012.

[4] Brathwaite, B. & Schreiber, I. (2009), *Challenges for game designers*, Course Technology.

[5] Österlind, D. (2009), 'Interactive storytelling.', http://www.davidosterlind.com/wp-content/uploads/2010/07/Essay_David%C3%96sterlind.pdf. Last accessed 27 December 2013.

[6] A.L.I.C.E. Artificial Intelligence Foundation (2012), 'A.l.i.c.e. program downloads', http://www.alicebot.org/downloads/programs.html. Last accessed 27 December 2013.

[7] Salen, K. & Zimmerman, E. (2003), *Rules of play: Game design fundamentals*, MIT press.

[8] Mateas, M. & Stern, A. (2003), 'Façade: An experiment in building a fully-realized interactive drama', in 'Game Developers Conference, Game Design track', p. 82.

[9] Mateas, M. & Sengers, P. (1999), 'Narrative intelligence', in 'Proceedings AAAI Fall Symposium on Narrative Intelligence', pp. 1–10.

[10] Mateas, M. (1999), 'An oz-centric review of interactive drama and believable agents', *Artificial intelligence today*, pp. 297–328.

[11] Mateas, M. & Stern, A. (2002), 'A behavior language for story-based believable agents', *Intelligent Systems, IEEE*, vol. 17, no. 4, pp. 39–47.

[12] Mateas, M. & Stern, A. (2005), 'Structuring content in the façade interactive drama architecture', in 'Proceedings of Artificial Intelligence and Interactive Digital Entertainment', pp. 93–98.

[13] Wardrip-Fruin, N., Mateas, M., Dow, S. & Sali, S. (2009), 'Agency reconsidered', *Breaking New Ground: Innovation in Games, Play, Practice and Theory. Proceedings of DiGRA 2009.*

[14] El Shazly, N. & Aly, S. (2011), 'Profile-based mood extraction from a social network', http://www.lidi.info.unlp.edu.ar/WorldComp2011-Mirror/ICM4290.pdf. Last accessed 27 December 2013.

[15] Liu, H., Lieberman, H. & Selker, T. (2003), 'A model of textual affect sensing using real-world knowledge', in 'Proceedings of the 8th international conference on Intelligent user interfaces', ACM, pp. 125–132.

[16] Thelwall, M., Buckley, K., Paltoglou, G., Cai, D. & Kappas, A. (2010), 'Sentiment strength detection in short informal text', *Journal of the American Society for Information Science and Technology*, vol. 61, no. 12, pp. 2544–2558.

[17] Zhang, L., Barnden, J., Hendley, R., Lee, M., Wallington, A. & Wen, Z. (2008), 'Affect detection and metaphor in e-drama', *International Journal of Continuing Engineering Education and Life Long Learning*, vol. 18, no. 2, pp. 234–252.

[18] Jung, Y., Park, H. & Myaeng, S. (2006), 'A hybrid mood classification approach for blog text', *PRICAI 2006: Trends in Artificial Intelligence*, pp. 1099–1103.

[19] Liu, H., Lieberman, H. & Selker, T. (2002), 'Automatic affective feedback in an email browser', in 'In MIT Media Lab Software Agents Group', ACM.

[20] Bradley, M. & Lang, P. (1999), 'Affective norms for english words (anew): Instruction manual and affective ratings', *University of Florida: The Center for Research in Psychophysiology.*

[21] Tausczik, Y. & Pennebaker, J. (2010), 'The psychological meaning of words: Liwc and computerized text analysis methods', *Journal of Language and Social Psychology*, vol. 29, no. 1, pp. 24–54.

[22] Liu, H. & Singh, P. (2004), 'Conceptneta practical commonsense reasoning tool-kit', *BT technology journal*, vol. 22, no. 4, pp. 211–226.

[23] id Software (2011), 'Rage official site', http://www.rage.com/en. Last accessed 27 December 2013.

[24] Bioware (2007), 'Mass Effect Special Edition Documentary', Mass Effect Special Edition Xbox®video game system Game.

[25] Mateas, M. & Stern, A. (2003), 'Integrating plot, character and natural language processing in the interactive drama façade', in 'Proceedings of the 1st International Conference on Technologies for Interactive Digital Storytelling and Entertainment', TIDSE-03.

[26] McCoy, J., Treanor, M., Samuel, B., Reed, A., Wardrip-Fruin, N. & Mateas, M. (2012), 'Prom week', in 'Proceedings of the International Conference on the Foundations of Digital Games', ACM, pp. 235–237.

[27] Spierling, U., Weiß, S. & Müller, W. (2006), 'Towards accessible authoring tools for interactive storytelling', *Technologies for Interactive Digital Storytelling and Entertainment*, pp. 169–180.

[28] A.L.I.C.E. Artificial Intelligence Foundation (2012), 'Aiml: Artificial intelligence markup language', http://www.alicebot.org/aiml.html. Last accessed 27 December 2013.

[29] Ekman, P. (1992), 'An argument for basic emotions', *Cognition & Emotion*, vol. 6, no. 3-4, pp. 169–200.

[30] Hendrikx, M., Meijer, S., Van der Velden, J. & Iosup, A. (2011), 'Procedural content generation for games: A survey', *ACM Transactions on Multimedia Computing, Communications and Applications*.

[31] Holland, J. (2000), *Emergence: From chaos to order*, Oxford University Press.

[32] Stratton, C. (2003), 'Pyaiml (a.k.a program y)', http://pyaiml.sourceforge.net/. Last accessed 27 December 2013.

[33] Van Vliet, H. (2008), *Software engineering: principles and practice*, Citeseer.

[34] Maierhofer Software (2009), 'NHunspell: Free Spell-Checker, Hyphenation and Thesaurus for .NET', http://nhunspell.sourceforge.net/. Last accessed 27 December 2013.

[35] Pickrel, P. (1988), 'Flat and round characters reconsidered', *The Journal of narrative technique*, vol. 18, no. 3, pp. 181–198.

[36] NCTE/IRA (2010), 'List of personality traits', http://elenaist.edublogs.org/files/2010/11/Personality-Traits-List-of-Adj-1r20uec.JPG. Last accessed 27 December 2013.

[37] Duckworth, T. (1998), 'A dictionary of slang', http://www.peevish.co.uk/slang/. Last accessed 27 December 2013.

[38] The American's guide to speaking British (2000), 'The Best of British - Slang', www.effingpot.com/slang.shtml. Last accessed 27 December 2013.

[39] Allen, I.E. & Seaman, C.A. (2007), 'Likert scales and data analyses', *Quality Progress*, vol. 40, no. 7, pp. 64–65.

[40] Drachen, A. & Canossa, A. (2009), 'Towards gameplay analysis via gameplay metrics', in 'Proceedings of the 13th International MindTrek Conference: Everyday Life in the Ubiquitous Era', ACM, pp. 202–209.

[41] Tychsen, A. (2008), 'Crafting user experience via game metrics analysis', in 'Proceedings of the Workshop Research Goals and Strategies for Studying User Experience and Emotion at the 5th Nordic Conference on Human-computer interaction: building bridges (NordiCHI), Lund, Sweden', pp. 20–22.

[42] Epic Games (2010), 'Unreal Kismet User Guide', http://udn.epicgames.com/Three/KismetUserGuide.html. Last accessed 27 December 2013.

Appendix A

Sample conversation

An excerpt of a conversation between an NPC and a human tester is presented here. This was gathered during testing (Chapter 6). Included in the presented conversation are indications of mood changes and NPC RNF responses, indicated by parentheses.

A.1 Conversation with Vincent

Player: Hi
Vincent: HELLO

Player: What's your name?
Vincent: I AM VINCENT JACKSON

Player:How are you?
Vincent: I AM EXTREMELY PATIENT BUT THAT'S AN INAPPROPRIATE THING TO SAY (No response)

Player: Okay
Vincent: INDEED... ASK ME SOME RELEVENT. (No response)

Player: What do you want to achieve?
Vincent: IF I COULD I WOULD TRADE ALL MY MONEY TO BE HUMAN AGAIN. I WANT TO EXPERIENCE SUNLIGHT AND I WISH I COULD EAT GARLIC.

Player: Who are your allies?
Vincent: WHY YOU (SIGH)... I AM LOYAL TO MY FRIENDS. I DESPISE THE

PETTY SO-CALLED PURE BLOODS AND THIS HUNTER FELLOW.

(Mood changed to neutral)

Player: What are pure bloods?
Vincent: A PURE-BLOOD IS A VAMPIRE THAT WAS BORN A VAMPIRE.

Player: What is Hunter?
Vincent: A REAL PAIN IN MY NECK. HE WIPES OUT ALL VAMPIRES, TURNED OR PURE-BLOOD, KILLER OR HARMLESS. HE KILLS WITHOUT QUESTION OR MERCY. TO HIM A VAMPIRE IS JUST A CORPSE.

Player: What are the turned?
Vincent: HMMM... THE TURNED ARE PEOPLE THAT SURVIVE A VAMPIRE BITE AND BECOME A VAMPIRE THEMSELVES.

Player: Are you a vampire?
Vincent: (SIGH)... YES I AM ONE OF THE DAMNED

Player: What must I do?
Vincent: CAPTURE HUNTER...

Player: What is my mission?
Vincent: CAPTURE HUNTER...

(Mood changed to negative)

Player: What shall I do?
Vincent: (SIGH)... HELP ME FINISH THIS SERUM THAT'LL WIPE OUT ALL THE SO CALLED PURE. IMAGINE A FUTURE WHERE OUR PEOPLE ARE FREE AND WE NO LONGER HAVE TO ANSWER TO CAECUS OR ALEXANDER.

Player: Who is Caecus?
Vincent: JUST A PURE-BLOOD. SHE KILLED MY FAMILY AND TURNED ME. AND SHE'S A THORN IN MY SIDE BECAUSE AS THE RULER OF THE HOUSE OF STEINER SHE HAS POWER OVER US. I WILL HAVE MY VENGEANCE AGAINST HER.

Player: Who is Alexander?
Vincent: (SIGH)... SO CALLED RULER OF ALL THE VAMPIRES. HIS DEATH

SHALL COME SOON ... SIGH

Player: What is House of Steiner?
Vincent: OH.. THE HOUSE OF STEINER. A BUNCH OF ARROGANT PURE-BLOODS LEAD BY CAECUS. KILL HER AND THE HOUSE WILL PROBABLY CRUMBLE.

Player: What do you crave?
Vincent: WHY YOU (SIGH)... I LOVE MONEY BUT I LOVE BLOOD EVEN MORE. BUT I DO NOT BELIEVE IN KILLING PEOPLE, JUST ANIMALS.

Player: What are your fears?
Vincent: OH (SIGH) I FEAR DEATH, SINCE I AM DAMNED THERE IS NOTHING GOOD WAITING FOR ME ON THE OTHER SIDE.

Player: Who is your boss?
Vincent: TECHNICALLY, I ANSWER TO ALEXANDER. BUT THAT WILL SOON CHANGE.

Player: Who are your friends?
Vincent: I AM LOYAL TO MY FRIENDS. I DESPISE THE PETTY SO-CALLED PURE BLOODS AND THIS HUNTER FELLOW ... SIGH

Player: What is your job?
Vincent: HMMM... I RUN THE HOUSE OF JACKSON. WE TAKE IN THE NEWLY TURNED AND HELP THE FIGHT THEIR URGES BY FEEDING THEM ANIMALS' BLOOD ... SIGH

Player: Schooling?
Vincent: (SIGH)... THE HOUSE OF JACKSON LOOKS AFTER EVERYONE INNOCENT. (No response)

Player: School?
Vincent: I AM A HARVARD MAN, THROUGH AND THROUGH. IN MY OLD LIFE I WAS SUCCESSFUL LAWYER.

Player: What do you love?
Vincent: I LOVE MY TURNED BRETHREN LIKE FAMILY. I STILL LOVE SAMANTHA AND WISH TO BE WITH HER ONE DAY.

(Mood changed to positive)

Player: Who is Samantha?
Vincent: SHE WAS THE LOVE OF MY LIFE BUT CAECUS TOOK HER FROM ME AND TURNED ME INTO WHAT I AM NOW. I CANNOT BELIEVE THAT I USED TO WANT PEACE WITH THOSE PURE-BLOOD MONSTERS.

Player: Do you believe in God?
Vincent: INDEED... I BELIEVE THAT THERE MUST BE SOMETHING BETTER THAN THIS.